António Jesus Monteiro de Castro

A Distributed Approach to Integrated and Dynamic Disruption Management in Airline Operations Control

July 18, 2013

Supervisor: Professor Doutor Eugénio de Oliveira

Faculdade de Engenharia
Universidade do Porto
Rua Dr. Roberto Frias, s/n
4200-465 Porto, Portugal

Thesis submitted in partial fulfillment of the requirements for the Doctoral Degree in Informatics Engineering

This work was supported by Fundação para a Ciência e Tecnologia (FCT) through grant SFRH/BD/44109/2008.

Copyright © 2013 by António J. M. Castro

English text reviewed by Ricardo Tavares (ricardoltav@gmail.com)

Subtly,

you brought something new to my life.
I hardly realized what it was and its importance.
After all, it was just a distant star
that you decided to bring closer.
It seemed one among many, but it had a special feature:
when I'm near it, my heart glows with happiness.
I will keep it close,
hoping that you will never take it back from me.

António Castro

To the memory of my beloved sister, Laura Castro, who died in March, 17th., 2011 with just 40 years of age.

Acknowledgments

Life has two parts: the one we live and the one we do not live. I regret not to live the latter and not to fully live the former. I do not know what would have been the life I have not lived, but the unknown is always a good excuse for adventure and I would be thrilled to participate in it. However, I do know the life I have lived. The good things, the bad things and the missed opportunities.

During more than five years, a lot of things happened in my life. My divorce and the death of my sister are amongst those that had a great impact on my life. The two year battle against cancer, with all the ups and downs, some days lived with hope for a cure but a lot of other days confronted with the hard reality, turned those years into something very, very hard for all of us but, specially, for her. For me, that time was hard too, but seeing her in those last fifteen minutes of life, her breath, slow and increasingly spaced showing that she wanted to live, and that moment when she ceased to breathe, I still feel them today. In an instant, she was no longer among us and a huge void fills our heart and our mind starts to become full of unanswered questions. See you soon Laura!

Bad times also allow us to have good times. One I remember most happened the day before going to the funeral. For her it was nothing special but, for me, to have the company of someone who actually wanted to be there and feel the same as me, showed me that true friendship still exists. Even in the solitary dinner I had that day, she was there, listening and answering to my messages. You probably do not remember it but I do. Thanks!

To also see someone that I did not expect, in my sister's funeral, supporting me through that moment, is another example of true friendship. I know you remember it. Thanks!

These apparently small things, remind me that the important things are those that have a meaning for us, independently of being big or small.

This work was only possible thanks to the contributions of many people, who helped in several ways. It is hard to acknowledge all these people, whose true impact on the final work I may not even realize at this moment.

First, I would like to thank my parents. From the little they had, they gave me so much. The wisdom to know that it does not matter where we are from or if we come from a rich or poor family. It all depends on us, on what we value and on our choices.

Second, I would like to thank my daughters. Despite what they may think of me, they are my most precious legacy. I love you unconditionally.

Third, I would like to thank Paula, for all that you were, for all that you are and for all that you will be in my life.

Prof. Eugénio de Oliveira has been my supervisor and mentor since I decided to get a Master's degree in Artificial Intelligence and Intelligent Systems. His knowledge and friendship has been an important asset for me, not only during my PhD but, also, in my life. Please accept my deepest gratitude.

A word of appreciation goes to my advisory group. I am grateful to Prof. Pavão Martins and Prof. Luis Paulo Reis for reading and giving feedback regarding my work.

I also wish to thank Eng. Manoel Torres, TAP Portugal EVP, for believing in me and for all the support. The same gratitude extends to Captain Vasco Moura. I could not finish this work without your support. I am also grateful to Eng. António Aguiar for the help in reviewing the text.

Additionally, I would like to thank all my colleagues at LIACC/NIADR for their support, for the productive discussions and exchange of ideas, and also for the friendly environment they helped to create.

To Prof. Eugénio, Ana Paula, António Pereira, Henrique, Joana and Rosaldo, for all the happy moments around a table full of food, wine and cigars (I will never forget how to light a Cuban cigar).

To António Pedro Pereira, Bruno Aguiar, Francisca Teixeira, José Pedro Silva, José Torres and Leonardo Fraga for all the help during different stages of the prototype development.

To all the other colleagues at LIACC/NIADR (in alphabetic order): Andreia Malucelli, Célia, Gustavo, Jorge Teixeira, Luis Paulo Reis, Luis Sarmento and Pedro Brandão. Thanks for your company and help.

In TAP Portugal I have a fantastic team of colleagues and friends, that work with me in several information system projects (in alphabetic order): Carla Cunha, Cristina Serra, Gonçalo, Helena Patronilho, Isabel Girão, Jaime Crato, João Costa, João Paquete, Luís Barbosa, Nuno Vicente, Patrícia Manhão, Pedro Rica and Sandra Mendonça. Part of my work has also your contribution. I am also grateful to Marta for remembering me the importance of having a *sergeant* pushing for me all the time.

I am also grateful to Shawn Wolfe from NASA AMES Research Center, SISCOG and TAP Portugal. Your support was important in my dissertation work.

I also want to thank my ex-wife for all the support she gave me while we were married.

Finally, I would like to thank *Mother Nature* for creating a place called Caramulo. The mountain air, sunshine and mild breeze and the peaceful stillness of that rock in the middle of nowhere, inspired me and gave me the peace that I needed to write parts of this dissertation.

António Castro
May 2013

Resumo

As companhias aéreas fazem um enorme esforço para maximizar as suas receitas mantendo os seus custos no mínimo valor possível. Esta não é uma tarefa fácil e, por isso, as companhias aéreas investem em ferramentas que permitam otimizar o seu plano operacional. Infelizmente, qualquer plano operacional tem uma grande probabilidade de ser afetado, não só por grandes ruturas tal como a que aconteceu em Abril de 2010, devido à erupção do vulcão Islandês Eyjafjallajökull mas, mais frequentemente, por pequenas ruturas causadas por mau tempo, avarias nos aviões e absentismo dos tripulantes, por exemplo.

Estas ruturas afetam o plano original, atrasando os voos e causando aquilo a que se chama *Operação Irregular*. Existem estudos que estimam que as operações irregulares podem custar entre 2% a 3% da receita anual da companhia aérea e que, um melhor processo de recuperação pode resultar numa redução de custos de, pelo menos, 20% da sua operação irregular.

Os Centros de Controlo Operacional das companhias aéreas (CCO) tentam gerir estas ruturas procurando soluções com o mínimo de impacto no plano operacional e com o mínimo custo. Normalmente, o CCO resolve as ruturas de forma sequencial, i.e., primeiro resolve a parte correspondente ao recurso avião, depois a parte da tripulação e, finalmente, a parte dos passageiros. A maior parte do processo de resolução é feita manualmente por operadores experientes, tendo como ajuda algumas ferramentas computorizadas.

Nesta tese, estudamos o CCO da TAP Portugal assim como o trabalho proposto por outros investigadores nesta área, e propomos uma abordagem genérica *distribuída* e *descentralizada* para uma gestão de ruturas *integrada* e *dinâmica* no controlo operacional das companhias aéreas, baseada no paradigma dos Sistemas Multi-Agente (SMA). Esta é uma das contribuições principais do nosso trabalho

A abordagem é *distribuída* porque permite a distribuição funcional, espacial e física das várias funções e do ambiente (i.e., dos recursos disponíveis); é *descentralizada* porque algumas decisões são tomadas em nós diferentes da rede de agentes; é *integrada* porque inclui as principais dimensões do problema: avião, tripulação e passageiros; e é *dinâmica* porque, em tempo real, vários agentes estão a funcionar no ambiente reagindo às mudanças constantes.

Apesar de termos usado o CCO da TAP Portugal como caso de estudo, o nosso trabalho foi realizado com um pensamento *out-of-the-box*. Por pensamento *out-of-the-box*, queremos dizer que tentamos pensar para além dos requisitos do problema em concreto que estamos a resolver neste momento, explorando direções divergentes e envolvendo vários aspetos que, no futuro, podem ser uma mais-valia.

Propomos (e esta é mais uma das principais contribuições do nosso trabalho) um protocolo de negociação chamado *Generic Q-Negotiation (GQN)* para ser usado como mecanismo de decisão, no processo de gestão de ruturas. Este protocolo tem características que estão para lá daquilo que necessitamos para resolver o problema como o conhecemos hoje em dia. No entanto, estas características, para além de resolverem o problema atualmente, permitem que seja aplicado em futuros cenários diferentes, seja no mesmo domínio de aplicação seja em outros domínios de aplicação com características semelhantes.

A mesma linha de pensamento foi seguida para as outras contribuições do nosso trabalho. Por exemplo, na proposta de uma metodologia Agent-Oriented Software Engineering (AOSE) chamada *PORTO*, que resulta do uso e melhoramento de outras metodologias.

Baseado na nossa abordagem SMA e no protocolo GQN, realizamos várias experiências, comparando os resultados com o processo manual feito pelo CCO da TAP e, também, com um processo automático que implementa a típica abordagem sequencial seguida pelos CCOs.

Os resultados demonstram que a nossa proposta não só corrobora os estudos existentes relativos à possível redução de custos que possam resultar de um melhor processo de gestão de ruturas, como, também, tem a possibilidade de reduzir custos e chegar a soluções que equilibram as três dimensões do problema: avião, tripulação e passageiros.

Porto, Maio de 2013 *António Castro*

Abstract

Airline companies make a huge effort to maximize their revenue while keeping their costs at a minimum. This is not an easy task and, because of that, airline companies invest in tools that allow to optimize their operational schedule. Unfortunately, any operational plan has a strong probability of being affected, not only by large disruptions like the one that happened in April 2010 due to the eruption of the Iceland Eyjafjallajökull volcano but, more frequently, by smaller daily disruptions caused by bad weather, aircraft malfunctions and crew absenteeism, for example.

These disruptions affect the original schedule plan, delaying the flights, and cause what is called an *Irregular Operation*. Studies have estimated that irregular operations can cost between 2% and 3% of the airlines' annual revenues and that a better recovery process could result in cost reductions of at least 20% of its irregular operations.

Considering the above, the Airline Operations Control Center (AOCC) tries to manage disruptions by trying to find a solution with a minimum impact to the airline schedule and with the minimum cost. Usually, the AOCC solves the disruptions in a sequential process, i.e., first solving the aircraft part of the problem, then the crew part and, finally, the passenger part. Most of the process is done manually by experienced operators with the help of some computer-based tools.

In this thesis, we have studied the AOCC of TAP Portugal as well as the work of other researchers in this field in order to propose a *distributed* and *decentralized* general approach to *integrated* and *dynamic* disruption management in airline operations control, based on the Multi-Agent System (MAS) paradigm. This is one of the main contributions of our work.

The approach is *distributed* because it allows the functional, spatial and physical distribution of the intervening agent roles and the environment (i.e., resources available); it is *decentralized* because some decisions are made in different nodes of the agents' network; it is *integrated* because it includes the main dimensions of the problem: aircraft, crew and passengers; and it is *dynamic* because, in real time, several agents are performing in the environment, reacting to constant change.

Although we use the TAP Portugal's AOCC as a case study, our work was performed with an *out-of-the-box* thinking. By *out-of-the-box* thinking we mean trying to think beyond the requirements of the specific problem we are solving as of this moment in time, exploring divergent directions and involving a variety of aspects that, at this moment in time might not be relevant but, in the future, might be an asset.

We propose (and this is another one of the main contributions of our work) a negotiation protocol called *Generic Q-Negotiation (GQN)* to be used as a decision mechanism in the disruption management process. This protocol has characteristics that are beyond what we need to solve the problem as it is today. However, these characteristics, besides solving the problem as we know it today, allow it to be applied to different scenarios that may arise in the future, either for the same application domain or to different application domains that share similar characteristics.

The same line of thinking was followed in the other contributions of our work, including the proposal of an Agent-Oriented Software Engineering (AOSE) methodology called *PORTO*, which resulted from the use and improvement of other methodologies.

Based on our MAS-based approach and on the GQN protocol, we have performed several experiments, comparing the results with those of the manual process followed by the AOCC operators

at TAP and with an automated process that implements the typical sequential approach followed by the AOCCs.

The results show that our proposal, not only corroborates existing studies regarding the possible cost reductions that could result from a better disruption management process but, also, gives the possibility of reaching solutions that balance the utility of the three dimensions of the problem: aircraft, crew and passengers.

Porto, May 2013 *António Castro*

Résumé

Les compagnies aériennes font un énorme effort pour maximiser leurs revenus tout en gardant les coûts au minimum. Ce n'est pas une tâche facile et, en conséquence, les compagnies aériennes investissent sur des outils qui permettent d'optimiser leur calendrier opérationnel. Malheureusement, tout le plan opérationnel a une forte probabilité d'être affecté, non seulement par des perturbations importantes comme celle qui s'est produite en Avril 2010 en raison des cendres du volcan islandais Eyjafjallajökull, mais, surtout, par les petites perturbations quotidiennes causées par le mauvais temps, les dysfonctionnements d'avions et l'absentéisme de l'équipage, par exemple.

Ces perturbations affectent le plan calendrier initial, ce qui retarde les vols et provoque ce qu'on appelle une *opération irrégulière*. Des études ont estimé que des opérations irrégulières peuvent coûter entre 2% et 3% du chiffre d'affaires annuel aérien et qu'un meilleur procès de récupération pourrait se traduire par des réductions de coûts d'au moins 20%.

Considérant toutes les raisons antérieures, le Centre de contrôle des opérations aériennes (CCOA) essaye d'administrer les perturbations avec le minimum d'impact sur le calendrier de la compagnie et avec un coût minimum. Habituellement, le CCOA résout les perturbations dans un procès séquentiel, c'est à dire, en premier cas, la partie du probléme reliée aux avions, en suite, la partie du probléme regardant l'équipage et, finalement, la partie des passagers. La plupart du procès se fait manuellement par des opérateurs expérimentés avec l'aide de certains outils informatiques.

Dans cette thèse, nous avons étudié le CCOA de TAP Portugal ainsi que le travail d'autres chercheurs dans ce domaine et nous proposons une approche *distribuée* et *décentralisée* générale à la perturbation *intégrée* et *dynamique* de la gestion des opérations aériennes dans le contrôle, basé sur le paradigme du Système Multi-Agents (SMA). C'est l'une des principales contributions de notre travail.

L'approche est *distribuée* car elle permet la distribution fonctionnelle, spatiale et physique des rôles et de l'environnement (ressources disponibles); elle est *décentralisée* parce que certaines décisions sont prises à différents noeuds du réseau des agents; elle est *intégrée* car elle inclut les principales dimensions du problème: les avions, les équipages et les passagers; et elle est *dynamique* car, en temps réel, plusieurs agents sont capables de réagir à l'évolution constante de l'environnement.

Quoiqu'ayant utilisé le CCOA de TAP Portugal comme étude de cas, notre travail a été effectué avec une pensée *out-of-the-box*. Quand nous parlons de l´expression *out-of-the-box* nous voulons dire que nous essayons de penser au-delà des exigences du problème spécifique que nous vérifions en ce moment là. Notre intérêt est aussi d'exploiter les directions divergentes et diverses qui peuvent se révéler un atout, à l'avenir.

Nous proposons (et ceci est une autre des principales contributions de notre travail) un protocole de négociation appelé *Négociation-Q Générique (GQN)* pour être utilisé comme un mécanisme de décision dans le procès de gestion des perturbations. Ce protocole a des caractéristiques qui sont au-delà des besoins pour résoudre le problème tel qu'il se pose aujourd'hui. Toutefois, ces caractéristiques, en plus de résoudre le problème comme nous le connaissons, permettent qu'il soit appliqué dans les différents scénarios qui pourraient apparaître à l'avenir, soit pour le même domaine

d'application ou pour les différents domaines d'application qui partagent des caractéristiques similaires.

La même ligne de pensée a été suivie pour les autres contributions de notre travail, et inclue la proposition d'une méthodologie de Génie du Logiciel (AOSE), appelée *PORTO*, que dérive de l'utilisation et du progrès d'autres méthodes.

Sur la base de notre approche axée sur le SMA et sur le protocole GQN, nous avons effectué plusieurs expériences, en comparant les résultats avec le procès manuel effectué par les opérateurs AOCC de TAP et avec un procès automatisé qui met en oeuvre l'approche séquentielle classique suivie par les CCOAs.

Les résultats montrent que notre proposition, non seulement justifie les études existantes en ce qui concerne les réductions de coûts possibles qui pourraient résulter d'un procès de meilleure gestion des perturbations, mais, nous conduisent aussi à la possibilité de réduire les coûts et d'atteindre des solutions qui concilient l'utilité des trois dimensions du problème: les avions, les équipages et les passagers.

Porto, Mai 2013 *António Castro*

Contents

Acronyms

ACARS	**Aircraft Communications Addressing and Reporting System**: A digital datalink system for transmission of short, relatively simple messages between aircraft and ground stations via radio or satellite.
ACMI	**Aircraft Crew Maintenance Insurance**: Also known as Wet Lease. The leasing of an aircraft including the crew members, maintenance and insurance.
AEA	**Association of European Airlines**: An association that includes 32 major airlines and that has been the voice of the European airline industry for over 50 years.
AMS	**Agent Management Services**: The core agent which keeps track of all JADE programs and agents in the system.
AOCC	**Airline Operations Control Center**: An airline entity responsible for monitoring and problem solving during the execution of the airline plan.
AOCP	**Airline Operations Control Problem**: The daily process of solving unexpected events that might disrupt the airline schedule (or plan) with the minimum cost and according to specific rules. See also Disruption Management below.
AOSE	**Agent Oriented Software Engineering**: A methodological approach for the development of software oriented or based on software agents.
ASAS	**Automatic or Semi-Automatic Systems**: A software system that replaces the functional part of an entity by computerized programs that work autonomously. In an automatic system, decision making is also undertaken by the system. In a semi-automatic system, the final decision is made by the human operator.
ASP	**Airline Scheduling Problem**: The process of creating the airline schedule (or plan) that covers all of the airline network, maximizing the revenue and minimizing the costs related to aircraft and crew members, in a specific period.
ATA	**Actual Time of Arrival**: The actual time of arrival of a flight.
ATC	**Air Traffic Control**: A service provided by ground-based controllers who direct aircraft on the ground (at airports for take-off and landing) and in the air.
ATD	**Actual Time of Departure**: The actual time of departure of a flight.
CDM	**Collaborative Decision Making**: Information about the aircraft/flight movement in several airports. It helps in making decisions.
CDSP	**Cooperative Distributed Problem Solving**: A process of solving problems in a distributed way (physical, functional or spatial) in an environment where the entities are willing to cooperate, either because they have the same goal or because they are not able to solve the problem entirely by themselves.
CFMU	**Central Flow Management Unit**: A tool from EUROCONTROL that provides information about ATC slots.
DBQS	**Database Query System**: A system that allows a human operator to query a database and get information.
DF	**Directory Facility**: A yellow page service in JADE, where agents can publish their services.
DM	**Disruption Management**: A dynamic process of solving disruptions that affect an existing plan, in such a way that the impact and costs are minimized and, at the

	same time, complying with the required rules. See also Airline Operations Control Problem above.
DSS	**Decision Support System**: A computer-based information system that supports business or organizational decision-making activities.
EFB	**Electronic Flight Bag**: An electronic information management device that helps flight crews perform flight management tasks more easily and efficiently, in a paperless environment.
ETA	**Estimated Time of Arrival**: An estimated time of arrival for a flight.
ETD	**Estimated Time of Departure**: An estimated time of departure for a flight.
FAA	**Federal Aviation Administration**: An agency of the United States Department of Transportation with authority to regulate and oversee all aspects of civil aviation in the U.S.
GDP	**Gross Domestic Product**: The market value of all officially recognized final goods and services produced within a country in a given period of time.
HCC	**Hub Control Center**: An entity that some airlines have at their major or central airports (hubs) to help control the incoming and outgoing airline traffic. This entity typically exists when the airline operates a hub-and-spoke network.
HUB	**Hub-and-Spoke Network**: A system of connections arranged like a chariot wheel, in which all traffic moves along spokes connected to the hub at the center. Medium to large airline companies use this kind of network as a way of making a more efficient use of transportation resources. For example, aircraft are more likely to fly at full capacity, and can often fly routes more than once a day.
IATA	**International Air Transport Association**: IATA represents some 240 airlines comprising 84% of scheduled international air traffic. It is present in over 150 countries and has 101 offices around the globe.
IR	**Integrated Recovery**: A process that is able to recover all problem dimensions separately (not simultaneously). Usually sequentially and, as such, having a solving order.
MAS	**Multi-Agent System**: A software system composed of multiple interacting (intelligent) software agents.
MASDIMA	**Multi-Agent System for Disruption Management**: The advanced MAS prototype developed during this thesis that implements the approaches proposed by us.
MCS	**Movement Control System**: A software system to control the flight and aircraft information related to departures, takeoff, landing and arrival, amongst other information.
MTOW	**Maximum Takeoff Weight**: The maximum weight at which the pilot of the aircraft is allowed to attempt to take off, due to structural or other limitations.
MVT	**Aircraft Movement Message**: A message that includes information about the movement of an aircraft/flight. Typically the OOOI information.
NB	**Narrow Body**: An aircraft with a single aisle. Typically used to perform short to medium-range flights.
NOTAM	**Notice to Airmen**: Notifications to aircraft pilots of any hazards en route or at a specific location. The NOTAMs are usually provided by the aviation authority.

OOOI	**Out, Off, On and In**: For every flight/aircraft there are four important times to register: Out of gate time (gate departure), Off ground time (takeoff), On ground time (landing) and In gate time (gate arrival).
OR	**Operations Research**: Mathematical or scientific analysis of a process or operation, used for making decisions.
PIL	**Passenger Information List**: A list with the names, seats and other relevant information of the passengers on board of a flight.
PIR	**Partial Integrated Recovery**: A process that is able to recover at least two, but not all, of the problem dimensions, simultaneously or not.
RMA	**Remote Management Agent**: The agent in JADE which handles the GUI interface
SEF	**Portuguese Immigration Services**: The service responsible for give effect to the policy of immigration and asylum in Portugal, according to the provisions of the Constitution and the law and government guidelines.
SIR	**Simultaneously Integrated Recovery**: A process that is able to recover all problem dimensions simultaneously. Here, all dimensions are of equal importance since there is no solving order.
SITA	**Société Internationale de Télécommunications Aéronautiques**: Provider of global information and telecommunication solutions for the air transport industry.
STA	**Schedule Time of Arrival**: The original arrival time of a flight.
STD	**Schedule Time of Departure**: The original departure time of a flight.
WB	**Wide Body**: An aircraft with two passenger aisles, also known as a twin-aisle aircraft. Typically used to perform long-range flights.
ULD	**Unit Load Device**: A pallet or container used to load luggage, freight, and mail on a aircraft.
UML	**Unified Modeling Language**: A standardized general-purpose modeling language that is used in software engineering. It includes a set of graphic notation techniques to create visual models of the systems being modeled.

Nomenclature

Sets

A	Set of attribute names of a dimension.
Ac	Set of aircraft in a specific time.
$Airp$	Set of airports.
As	Set of activities in a specific time.
CA	Set of attributes of a competence.
CD	Set of attribute domains of a competence.
CS	Set of candidate solutions of a respondent agent.
Cw	Set of crew members in a specific period of time.
Edg	Set of flights in graph G.
Fl	Set of flights in a specific period of time.
I	Set of attribute domains of a dimension.
J	Set of negotiation attributes.
L^o	An ordered set of negotiation messages in $NegP^o$ (OA negotiation process).
L^{Ra}	An ordered set of negotiation messages in $NegP^{Ra}$ (RA negotiation process).
O	Set of organizer agents (in a negotiation).
PA	Set of problem domain attributes.
Pxd	Set of disrupted passengers in a flight.
R	Set of respondent agents (in a negotiation).
R_a	Set of respondent agents in a negotiation $NegP^o$ (organizer agent negotiation process.
R_b	Set of respondent agents in a negotiation $NegP^{Ra}$ (respondent agent negotiation process.
RT	Set of restrictions.
Sac	A Solution Set for the aircraft problem as seen by the *aircraft specialist* agent.
Scw	A Solution Set for the crew problem as seen by the *crew specialist* agent.
Spx	A Solution Set for the passenger problem as seen by the *passenger specialist* agent.
V	Set of attribute score functions of a dimension.
Vrt	Set of airports in graph G.
VP	Set of preferred attribute values of a dimension.

Tuples

AT	A q-learning action.
AV	Partial-Solution attribute values.
C	Competence of an agent.
d	Dimension of a problem.
F	Feedback for each attribute of each dimension.
G	A graph used by the *passenger specialist* agent.
IP	Interaction Protocol in the negotiation model NegMod.
$NegMod$	Our Negotiation Model.
$NegP^o$	A negotiation process from the point of view of an organizer agent.
$NegP^{Ra}$	A negotiation process from the point of view of an organizer agent.
P	Problem to be solved.

ps	Partial-solution.
Q	State, action and Q-Value of the Q-learning algorithm.
S	Solution to a problem.
ST	A q-learning state.

Elements of Sets and Tuples or Indexes

a	An agent (in general).
ac	A specific element of the set *Ac*.
as	A specific element of the set *As*.
at	An action, i.e., an element of the n-tuple *AT*.
b	Another agent (in general).
cw	A specific element of the set *Cw*.
f	A feedback.
fl	A specific element of the set *Fl*.
i	Index of a dimension.
j	An index of an element of a set or tuple; An issue or attribute.
o	An organizer agent.
pa	An attribute (in general) from the problem domain.
pxd	A specific element of the set *Pxd*.
r	A respondent agent.
rt	A restriction.
sac	A specific element of the set *Sac*.
scw	A specific element of the set *Scw*.
spx	A specific element of the set *Spx*.
st	A state, i.e., an element of the n-tuple *ST*.

Variables

ad	A disrupted aircraft included in the set *Ac*.
asd	A disrupted activity included in the set *As*.
c	A comment by the organizer agent to a proposal presented by a respondent agent.
cwd	A disrupted crew member included in the set *Cw*.
ev	Evaluation assigned by the organizer agent to a proposal.
fd	A disrupted flight included in the set *Fl*.
id	An identifier of something.
m	Total number of attributes on a partial-solution; Total number of attributes on a dimension; Total number of respondent agents.
n	Total number of dimensions in a problem; Total number of partial-solutions in a solution; Total number of organizer agents.
p	Total number of proposals received; Total number of proposals sent.
rw	A reward (value of) in the q-learning algorithm.
s	Total number of requests and informs received.
t	An instance of time or a round in a negotiation.
uv	Utility value of a proposal/solution for the agent that sent it..
V^P	Value of an offer according to a score function.
V^i_j	The score value of issue *j* in dimension *i*.
V^t_o	Evaluation made by agent *o* does at round *t*.

w Total number of attributes (in general).
W_j^i Weight of issue j in dimension i.
α_i Weight of dimension i.

Miscellaneous

CL	Communication language of a negotiation model.
CV	A scoring function that evaluates the preferred solutions (partial) according to the competence of an agent.
DL	Domain language of a negotiation model.
E	Environment of a negotiation model.
RF	A List of request and inform tuples received during the inter-respondent agents negotiation.
$O^t_{r\to o}$	Offer from a respondent agent r to an organizer agent o at round t; vector of values proposed by agent r to agent o at round t.
$O^t_{r\to o}[j]$	Value of attribute j presented by agent r to agent o at round t.
$Q(st,at)$	The Q-value (in the q-learning algorithm) that corresponds to the execution of action at when at state st.
RI	Rules of interaction of an IP (Interaction Protocol).
$Rw_r^{t+1}(O^t_{r\to o})$	A function that calculates a reward (q-learning) that agent r receives at round $t+1$ after presenting a proposal O to agent o in the previous round t.
SM	Sequence of messages exchanged during a negotiation process.
$U_o(O^t_{r\to o})$	Utility of offer o for an organizer agent o.
V^i	Score function for dimension i.

Air Transport Metrics

$\overline{CC}$	Average of Crew Costs in m.u.
$\overline{CCrcv}$	Average of Crew Cost Recovery Ratio.
$\overline{CCmin}$	Average of Crew Cost per Minute of the original problem flight delay.
$\overline{CwD}$	Average of Crew Delays in minutes.
$\overline{FC}$	Average of Aircraft and Flight Costs in m.u.
$\overline{FCrcv}$	Average of Flight Cost Recovery Ratio.
$\overline{FCmin}$	Average of Flight Cost per Minute of the original problem flight delay.
$\overline{FD}$	Average of Flight Departure Delays in minutes.
$\overline{FD15min}$	Average Number of Flights with Departure Delays greater than 15 minutes.
$p\left(\overline{FD15min}\right)$	Percentage of the Number of Flights with Departure Delays greater than 15 minutes.
$\overline{FDrcv}$	Average of Flight Departure Delay Recovery Ratio.
$\overline{PC}$	Average of Passenger Costs in m.u.
$\overline{PCmin}$	Average of Passenger Cost per Minute of the original problem flight delay.
$\overline{PD}$	Average of Passenger Trip Time Delays in minutes.

Negotiation Outcome Metrics

GF	Global Fairness of the negotiation winner solutions.
$\overline{Ua/c}$	Average of the Aircraft partial-solutions Utility of the Negotiation Winner Solutions for the Aircraft Agent.

$\overline{Ucrew}$ Average of the Crew partial-solutions Utility of the Negotiation Winner Solutions for the Crew Agent.

$\overline{Upax}$ Average of the Passenger partial-solutions Utility of the Negotiation Winner Solutions for the passenger Agent.

$\overline{Usup}$ Average of the Negotiation Winner Solutions Utility for the Supervisor Agent.

$\overline{Usw}$ Average Utilitarian Social Welfare of the Negotiation Winner Solutions for the Supervisor Agent.

Protocol Performance Metrics

$\overline{Mprb}$ Average Number of Messages per Problem, exchanged by agents during the negotiation.

$\overline{Msg}$ Average Number of Messages Exchanged by agents during the negotiation.

$\overline{NR}$ Average Number of Negotiation Rounds to Reach an Agreement.

$\overline{NT}$ Average Negotiation Time.

$\overline{ST}$ Average Negotiation Search Time.

Solution Quality Metrics

$\overline{A/Cact_{ei}}$ Arithmetic Average of the percentage of times that an action i was used to solve the aircraft part of a problem in the experimental run e.

$A/Cqual$ Aircraft Solution Quality.

$\overline{CRWact_{ei}}$ Arithmetic Average of the percentage of times that an action i was used to solve the crew part of a problem in the experimental run e.

$CRWqual$ Crew Solution Quality.

$\overline{PAXact_{ei}}$ Arithmetic Average of the percentage of times that an action i was used to solve the passenger part of a problem in the experimental run e.

$PAXqual$ Passenger Solution Quality.

$ITGqual$ Integrated Solution Quality, i.e., the quality of the solution that includes the three dimensions of the problem: aircraft, crew and passenger.

List of Figures

List of Tables

Chapter 1
Introduction

Abstract The main goal of this chapter is to introduce the research methodology we have adopted in this thesis, as well as, a brief description of our object of study and the motivation behind it. As part of the research methodology we present the observations we have made as well as the hypotheses we formulated based on those observations, including the results we expect to obtain by proving the hypotheses. The main contributions of our work are presented and we conclude the chapter with the structure of this thesis.

1.1 Overview

Regarding *research methodologies* and considering the *motivation* behind them, it is possible to say that there are two main ones (Oliveira, 2010):

- *Pure* or *Basic* research: the main motivation is to expand man's knowledge, not to create some practical system. It contributes to a more theoretical and abstract formulation of phenomena, i.e, enhances the understanding of phenomena.
- *Instrumentalist* research: It includes the perspective that a scientific theory is a useful instrument in understanding the world. Contributes to make the human intervention more effective in real world environments.

The *instrumentalist* research can be further divided in two categories: *Applied* research and *Problem-Oriented* research. The former starts with a technology and applies it to processes, e.g, physical, organizational, social, etc. The latter, starts with the problem and seeks the most appropriate techniques to solve it.

In this thesis we follow the *problem-oriented* line of research, trying to have, at the same time, an *out-of-the-box* thinking. By *out-of-the-box* thinking we mean to try to think beyond the requirements of the specific problem we are solving as of this moment in time, exploring diverging directions and involving a variety of aspects that, at this moment in time might not be relevant but, in the future, might be an asset. For example, in Chapter 6 we present a negotiation protocol that has characteristics that are beyond what we need to solve the problem as it is today. However, these characteristics, besides solving the problem as we know it today, allow it to be applied in different scenarios that might appear in the future.

The problem we are trying to solve is that of *Disruption Management* in Airline Operations Control, i.e., to manage unexpected events that might affect flights, causing departure or arrival delays, minimizing delays and the costs involved.

Airline companies spend time and money in creating optimal operational plans to maximize the revenue. However, at the day of operation, the unpredictable events (bad weather, aircraft malfunctions and crew absenteeism, for example) might change completely that operational plan and, consequently, the revenue objectives behind it (Clausen *et al.*, 2010).

To deal with this problem, airline companies have a special department called *Airline Operations Control Center* (AOCC) that includes teams of human experts specialized in solving prob-

lems related to aircraft, crew members and passengers, in a process called disruption management. In this thesis, due to the professional experience of the author, we have observed and studied the AOCC of TAP Portugal, the major Portuguese airline company, and we use it as our case study scenario.

The Multi-Agent System (MAS) paradigm allows to model systems that include software agents, representing roles or functions as well as the surrounding environment, including interactions amongst the agents. This paradigm has assumed an increasing importance not only as a research paradigm but, also, as a practical one to solve real world problems .

From our personal knowledge and study about the AOCC and from the study we have performed regarding the MAS paradigm, we found that this paradigm has characteristics (Wooldridge, 2009a; Elamy, 2005) that will help to propose a *Distributed* and *Decentralized* approach to *Integrated* and *Dynamic* disruption management in airline operations control. The approach is *Distributed* because it allows the functional, spatial and physical distribution of the roles and the environment (i.e., resources available), it is *Decentralized* because some decisions are taken at different nodes of the agents' network, it is *Integrated* because it includes the main dimensions of the problem: aircraft, crew and passengers, and it is *Dynamic* because, in real time, several agents are performing in the environment reacting to constant change.

This chapter is organized as follows. Section 1.2 presents the motivation for our work. Section 1.3 presents the research methodology, including the observations, hypotheses and expected results. In Section 1.4 the main contributions that were achieved are presented and, finally, in Section 1.5 we outline the structure of this thesis.

1.2 Motivation

Air Transport has come a long way since the first world passenger airline, *DELAG (Deutsche Luftschiffahrts-Aktiengesellschaft, or German Airship Transportation Corporation Ltd)* was established in 1909 as a branch of the Zeppelin Company (Grossman, 2013). They use a form of airship, called *zeppelins*, named after their inventor. At the beginning, flights were sightseeing tours but in 1919 this company started to schedule service between Berlin and southern Germany. The flight took 4 to 9 hours, a huge improvement when compared to the 18 to 24 hours that the same journey took by train. This company made 103 flights and carried almost 2.500 passengers and employed the world's first flight attendant, Heinrich Kubis.

The world's first scheduled passenger airline using an *airplane* was *The St. Petersburg-Tampa Airboat Line* (Smithsonian National Air and Space Museum, 2013; Sharp, 2013), that started in the 1st of January 1914 and lasted only four months. The flight from St. Petersburg to Tampa took 23 minutes, 11 hours less than traveling by train. The airline transported 1.204 passengers during its short period of existence.

Nowadays, air transport is an essential part of the daily life of millions of people around the world. According to the Association of European Airlines (AEA, 2010) an average of 650 million passengers use air transport each year and the aviation sector employs directly 1.6 million people generating 2.9 million jobs. And this is in Europe alone. Worldwide and according to the IATA[1] 2012 Annual Review (Tyler, 2012), aviation transported around 2.8 billion passengers in 2011, directly employing 8.4 million people and supporting 56.6 million jobs globally.

[1] International Air Transport Association

Air transport also has an important contribution to other sectors of the economy. For example, and according to the AEA report (AEA, 2010), 3 million jobs in tourism at Europe are supported by air transport (34.5 million worldwide), contributing € 140 Billion to GDP[2] yearly. Additionally, 42% of tourists (51% worldwide), i.e., 80 million of people visiting Europe arrive by air, and the aircraft's occupancy rates reached 76% in 2009 (78% worldwide in 2011), far exceeding train and car load factors.

To support all this demand an extensive network of airports, airlines and aircraft needs to exist. According to IATA in 2010 this worldwide network was composed of 3.846 airports, 1.568 commercial airlines, 23.844 aircraft in commercial service and 26.7 million commercial flights.

Nowadays, one of the main challenges faced by this industry is related to flight delays and the costs these delays represent to the passengers and to the airlines. According to the Eurocontrol Delays report for 2011 (CODA, 2012), the average of departure flight delays per delayed flight in Europe was 28 minutes. As it is possible to see in Figure 1.1, only 43.4% of the flights were on-time in 2011 while almost 30% of the delayed flights are in the range of 5 to 30 minutes delay.

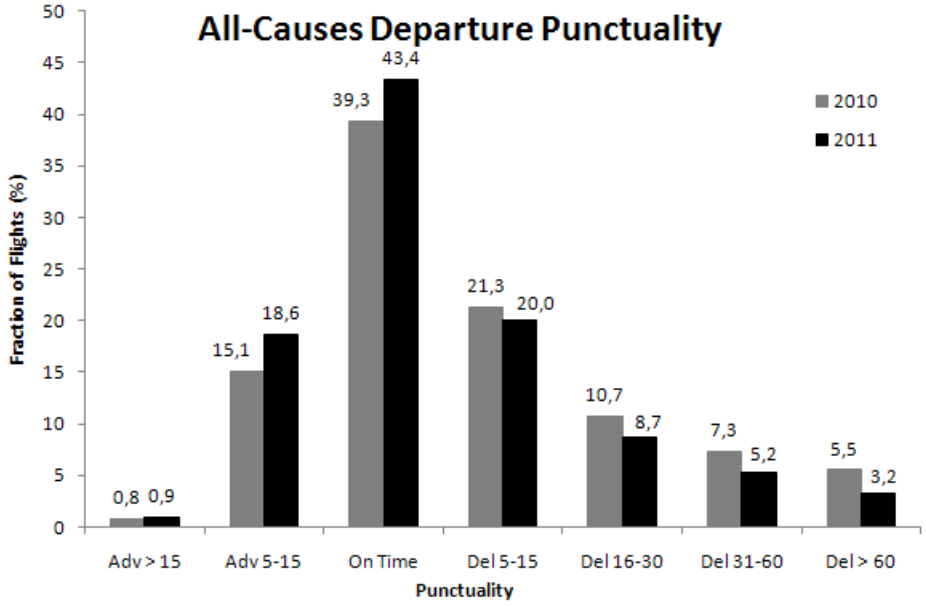


Fig. 1.1 Europe departure punctuality 2010 vs 2011

Regarding delays on arrival, the average was 29 minutes per delayed flight and only 26% of the flights arrived on-time. Besides the loss of *passenger-goodwill*[3] and the consequences that it has regarding the commercial image of the airline, some studies also calculate the cost of each minute of delay. For example, one of the best studies that were performed for the EUROCONTROL[4] by the Westminster University, known as the Westminster report (Cook *et al.*, 2004; Cook & Tanner, 2011), attributes a cost of € 72 for each minute of delay.

Airline companies face an important and difficult task in controlling their daily operations. Even for a small company like TAP Portugal[5] the size of the operational plan is considerable.

[2] Gross Domestic Product: market value of all goods and services produced within a country in a given period of time.

[3] The established reputation of a business regarded as a quantifiable asset.

[4] The European Organization for the Safety of Air Navigation. http://www.eurocontrol.int/

[5] http://www.flytap.com

TAP schedules 55 aircraft of 5 different types to 76 cities in 34 different countries, covering 1.850 weekly flights (approx. 8.000 flights per month) and assigns 3.132 crew members (composed of 821 pilots and 2.311 flight attendants) to those flights (TAP, 2011). Just for comparison, consider an airline like American Airlines[6] with 510 aircraft of 14 different types flying to 140 cities, with 2.700 daily flights and 25.000 crew members (Chen *et al.*, 2010). To operate such a system, airline companies use optimization techniques to be able to build their operational plan, maximizing the revenues and making an efficient use of resources (aircraft and crew members).

The optimal operational schedule that results from applying the optimization techniques has a strong probability of being affected, not only by large disruptions like the one that happened in April 2010 due to the eruption of the Iceland Eyjafjallajökull volcano but, more frequently, by smaller daily disruptions caused by bad weather, aircraft malfunctions and crew absenteeism, for example. These disruptions affect the originally scheduled plan, delaying the flights, and cause what is called an *Irregular Operation*. If nothing is done to manage the disruption, the delay can be propagated to other flights, making the problem more difficult to solve.

Studies have estimated that irregular operations can cost between 2% and 3% of the airline annual revenue (Chen *et al.*, 2010) and that a better recovery process could result in cost reductions of at least 20% of its irregular operations (Irrang, 1996). Consider the specific case of TAP Portugal, that according to the 2010 Annual Report (TAP, 2011), had an annual revenue of €1.986,3M. The irregular operations could have had an estimated cost between €39,7M to €59,5M. A better recovery process could thus mean a cost reduction of its irregular operations between €7,94M to €11,9M.

Considering the above information, we consider research on this domain to be very important and we hope to positively contribute to its improvement through the work we present in this thesis.

1.3 Research Methodology

As we stated in the *overview* section, in this thesis we follow the *problem-oriented* line of research. We start with the problem to solve, i.e., *disruption management in airline operations control*, and try to find the best methods, models or techniques to solve it.

There are many variations on the specifics of a scientific methodology. However, they all share common steps. We have adopted a research methodology that includes the following steps:

1. ***Identification of a problem***
   Considering the motivation as presented in Section 1.2 we identified the *disruption management in airline operations control* as the problem whose solution we want to contribute to.
2. ***Make observations about it***
   Starting from the identification of the problem we observed the Airline Operations Control Center (AOCC) of TAP Portugal. We have also studied the related literature to be aware of the work done by other researchers. A detailed description of the problem is presented in Chapter 4 and a list of the main observations is presented in Section 1.3.1.
3. ***Define one or more hypotheses including expected results***
   From the study we have performed and from the observations made, we stated some hypotheses that, we believe, will allow to propose methodologies (as well as techniques, algorithms and

[6] http://www.aa.com

tools) to better solve the existing problems. The hypotheses are presented in Section 1.3.2 and the expected results in Section 1.3.3.

4. ***Preparing the test laboratory***
   Having the observations, a detailed description of the problem and the hypotheses, we started to develop the advanced prototype that will help us to support our proposed solution. Our approach is presented in Chapter 7 and a description of the advanced prototype is presented in Appendix A.
5. ***Test the hypotheses by performing experiments***
   We have designed the experiments according to the hypotheses and we used the advanced prototype to perform the experiments. Section 8.1 presents the scenarios, metrics and approaches used.
6. ***Analyze the data and draw conclusions about the hypotheses***
   We collected data from the experiments and an interpretative analysis of the results was made. The results and a discussion about them are presented in Section 8.2. Having the results from the experiments and the hypotheses we were able to perform a critical analysis of our work, reaching conclusions about those hypotheses. The conclusions are presented in Chapter 9.

1.3.1 Observations

We have observed how the AOCC of TAP Portugal is organized and how it works, including the existing roles, functions and goals. To better understand how AOCC personnel solve problems, we have made interviews with them. Additionally, we have studied related literature to be aware of what other researchers are proposing regarding the problem we want to solve. Therefore, the following main observations result from this initial stage.

- ***First Observation***
  Small airline companies typically have only one operational base with aircraft and crew members, operating a point-to-point network of flights. Medium to large airline companies might have more than one operational base, each one with aircraft, crew members and flights. These companies typically have one *HUB*[7] and the largest ones might have more than one.
- ***Second Observation***
  Small to medium airlines have an AOCC with its roles in the same physical space. Large companies might have an AOCC plus additional HCC[8]. Typically the organization of the AOCCs is a multilevel hierarchy. There is an *Operations Control Supervisor* (OCS) that is the main authority regarding the operational control. At a second level there are teams of people, each team lead by a different manager, responsible for one of the different dimensions of the problem, i.e., aircraft, crew and passengers. There are also other teams related to supporting functions, e.g., maintenance and flight operations. At a third level, we have the members of each team, with the responsibility of finding solutions to the problems. There is a mix of cooperation (between elements of the third level) and authoritative control (between different levels). In order to monitor the operations, three different roles are needed: one for monitoring crew events, one for monitoring aircraft events and, finally, another one for monitoring passenger events. When an event

[7] Hub-and-Spoke Network: A system of connections in which all traffic moves along spokes connected to the hub at the center.
[8] Hub Control Center: An entity that some airlines have at their major or central airports (hubs) to help control the incoming and outgoing airline traffic.

is detected (flight delay, crew delay, passenger boarding problems) the person that detected it informs the other teams.

- ***Third Observation***
  Each team has its own goals to achieve, typically to minimize delays and costs related to its domain of expertise. The process of solving a flight delay is usually sequential: First, the team associated with the aircraft dimension finds a solution for the aircraft part problem; using this solution, the team associated with the crew dimension seeks a solution regarding the crew part; finally, the passenger part is solved by the team associated with the passenger dimension, considering the two previous solutions. This sequential order of problem solving might impose an order of importance to each part of the problem. Depending on when the problem is detected, there might be a time restriction to solve it. The final decision regarding to apply or not the solution is taken by the OCS. The OCS works like the representative of the airline, having global objectives regarding all the components of the solution. The implementation of the decision is carried out by each team according to their domain.
- ***Fourth Observation***
  Typically, the AOCC uses the airline's own resources to solve the problems, i.e., its own aircraft and crew members and tries to find solutions for the passengers using its own flights. However, sometimes, they use resources from other airline companies, for example, aircraft and crew members, typically known as *ACMI*[9]. In this case, the AOCC's preferences regarding the possible solutions should be kept private.
- ***Fifth Observation***
  Some of the roles perform functions that are very repetitive, for example, monitoring or finding solutions for the aircraft, crew members and passengers. The users apply knowledge that was transmitted by older colleagues and from the training they received, but also knowledge obtained from their own experience. Most of the time they use *rules-of-thumb*, a kind of hidden knowledge, to solve the problems. The environment, i.e., the information available, might change often and unexpectedly making it very dynamic.
- ***Sixth Observation***
  The information system available for the users in the AOCC that have to find solutions to the problems, provides only operational information, that is, information that is necessary for the operation to be performed (for example, aircraft status and maintenance, crew data related with licenses, qualifications and roster). Team members can have access to airport information or weather information, for example, but not in an integrated way. They need to use different computer systems for that.

1.3.2 Hypotheses

Having the observations we have formulated the following hypotheses.

- ***First Hypothesis***
  Based on the *first* and *second observation* we hypothesize that the *Multi-Agent Systems (MAS) paradigm* is appropriated as a modeling tool, encompassing the *spatial* and *physical distribution* identified in the first observation, i.e., to distribute the data and the roles. It also allows to achieve

[9] ACMI: Aircraft, Crew, Maintenance and Insurance. Airline companies rent the aircraft with crew members as well as maintenance services and insurance.

the *functional distribution* identified in the *second observation*, by distributing the roles by agents according to the dimensions of the problem, required expertise and contemplating the individual goals and preferences. This paradigm will also contribute to a flexible and scalable system as required by *both observations*, allowing the airline company to change its AOCC organization according to its size, needs and policy. Additionally, this paradigm has the potential to represent resources that do not belong to the airline as indicated by the *fourth observation*.

- ***Second Hypothesis***
  Based on the *third observation* and on part of the *second* we hypothesize that an *automatic negotiation protocol*, integrated in a MAS, will allow to achieve an integrated solution (one that includes the different parts of the problem) contributing to a more equal importance of each part as an opposition to the current sequential solving process that imposes an order of importance to each part. This can be achieved without compromising the goals of each team as required by the *third observation*. This negotiation protocol will also allow each agent to represent a part of the problem solving process, having its local or individual goals and preferences. At the same time, it can allow the agent representing the supervisor to have its global goals and preferences. This negotiation protocol can also allow to keep the preferences private, specially for the supervisor role, complying with the requirement that might emerge from the *fourth observation*. Finally, a negotiation protocol with a *learning mechanism* can contribute to partially achieve the requirements identified in the *fifth observation* regarding the dynamics of the environment.
- ***Third Hypothesis***
  Based on the *fifth observation* we hypothesize that having agents that implement different problem solving algorithms, according to the different expertise of the teams, can contribute to eliminate the repetitive tasks performed by the different roles and, at the same time, provide solutions that do not rely solely on the use of *rules-of-thumb*. This will increase not only the number of candidate-solutions to choose from but also, the quality of those candidate-solutions.
- ***Fourth Hypothesis***
  Based on the *sixth observation*, we hypothesize that by *agentifying* the different systems (e.g., weather forecast, flight boarding information, etc.) into the MAS we have proposed on the *first hypothesis*, we may be able to increase the quality of the candidate-solutions found by the problem solving agents referred to on the *third hypothesis*. At the same time, this will allow for the negotiation mechanism referred to on the *second hypothesis*, to reach better overall solutions.

1.3.3 Expected Results

To simplify and to better understand this section we assume the following terminology. The *organizer agent* is the one that represents the *supervisor role* and, as such, the *global interest of the airline* in the AOCC. The *manager agents* are the agents that *represent each part of the problem, i.e., a dimension*, having *local objectives*, *preferences* and *expertise*.

Considering the hypotheses we have formulated in the previous section we expect to get the following results.

- ***Regarding the first hypothesis*** we expect to obtain a *MAS Architecture* that allows to:

1. *Distribute roles* by agents according to the dimension of the problem (aircraft, crew and passenger) and required expertise (*functional distribution*), contemplating at the same time, the individual goals and preferences leading to decentralization (*autonomy and privacy*).
2. *Distribute data* by different databases (*spatial distribution*).
3. *Distribute the agents and data* by different computers and locations (*physical distribution*).
4. *Increase scalability* by adding, removing and changing roles, agents and environment according to the AOCC organization needs and policy.
5. *Represent resources or agents* that do not belong to the airline company.

- ***Regarding the second hypothesis*** we expect to obtain a *negotiation protocol* that:

1. *Increases the Social Welfare* (the sum of agent's individual utilities (Sandholm, 1999)) of the manager agents, allowing also the organizer agent to increase its utility. This will contribute to a better integrated solution.
2. *Balances the utility* of each manager agent, contributing to a more equal importance of each dimension of the problem without implying a decrease in the organizer agent's utility.
3. *Allows the autonomy and privacy* of the agents regarding its goals and preferences.
4. *Reaches an agreement in fewer rounds* (and, thus, reducing the time to get a solution) reacting, at the same time, to changes in the environment.
5. *Achieves all of the above without compromising the goals* of the organizer and manager agents.

- ***Regarding the third hypothesis*** we expect to have *problem solving algorithms* for each manager agent that:

1. *Increases the number of candidate-solutions* for each manager agent to choose from.
2. *Increases the quality of candidate-solutions* for each manager agent, i.e., candidate-solutions that could increase the utility for each manager.

- ***Regarding the fourth hypothesis*** we expect to *integrate different systems into the MAS* that:

1. *Bring additional information* that allows to improve the quality of candidate-solutions by increasing the utility for each manager agent (this expected result is closely related to the *third hypothesis* above).
2. *Bring other means of transportation*, allowing to have more diverse and alternate candidate-solutions and, as such, increase the number of candidate-solutions to choose from.
3. *By providing better and richer* candidate-solutions will enable the negotiation mechanism to reach better solutions, i.e., solutions that increase the utility of the organizer agent and the social welfare of the manager agents.

1.4 Main Contributions

The main scientific contributions of this thesis result from the study we have done of the organization of the Airline Operations Control Center (AOCC), as well as from the analysis, design and implementation we have done of the advanced prototype we have developed during our work. Therefore, we would like to point out the following contributions:

- **A specification of an *Agent-Oriented Methodology* (Chapter 5)**.
  To analyze, design and develop the advanced prototype we followed an agent-oriented methodology. We started by studying the *GAIA* methodology (Zambonelli *et al.*, 2003) but we found that it would be important to complement it with new phases and new and improved graphical notations. The improved methodology we propose, called *PORTO*, includes three new phases: *requirements* (based on another proposed methodology called *TROPOS* (Bresciani *et al.*, 2004)), *implementation* and *test and validation*. Additionally, *new* or *improved notations* were proposed based on the UML standard. A previous version of *PORTO* was used as the basis for the work of other researchers, such as (Silva *et al.*, 2012) and (Passos *et al.*, 2011).
- **A *Multi-Agent System Architecture* that represents the AOCC (Chapter 7)**.
  The AOCC is represented as an organization of agents, a multi-agent system, where the roles that correspond to the most frequent tasks are performed by intelligent agents. The human experts represented by agents, are able to interact with the intelligent agents by supervising the system and choosing the final decision out of the solutions proposed by the MAS. The architecture allows *functional*, *spatial* and *physical distribution*, promoting the *integration* of the three main dimensions that exist on the AOCC: aircraft, crew and passengers. Additionally, it allows a *dynamic* approach to the problem by solving it in real time, which implies an ever changing scenario. The design of the architecture was made with *out-of-the-box* thinking, supporting not only the current roles, functions and tools that exist on the AOCC but, also, *future tools* or *different organizations* as it is shown in Section 7.8. We also believe that the *architecture* is *generic* enough to be used in other application domains with characteristics similar to those of the AOCC.
- **A specification of a *Generic and Adaptive Negotiation Protocol* (Chapter 6)**.
  To be able to achieve an *integrated solution* where each dimension of the problem (represented by an agent) has an equal opportunity to be selected, we decided to use automated negotiation as a decision mechanism. We have defined and implemented a negotiation protocol called *GQN* - Generic Q-Negotiation. We call it *Generic* because it can be used in very different and heterogeneous environments. The *Q-Negotiation* part of the name comes from the fact that it inherits the characteristics of the Q-Negotiation Protocol (Rocha & Oliveira, 1999), an adaptive protocol that includes the Q-Learning algorithm. This protocol has characteristics that, in our opinion, allow it to be applied in more application domains (making it more generic). *GQN* is an adaptive protocol for multi-attribute negotiation with several rounds and qualitative feedback, with interdependent dimensions, supporting two types of agents, i.e. organizer and respondent agents, with full or partial knowledge.
- **A method to *Calculate the Quality Costs from the Passenger's Point of View* (Section 7.5.2)**.
  Typically, airline companies attribute a monetary value to each minute of flight delay, including a component for the loss of *passenger goodwill*. We introduce the concept of *Quality Operational Costs* of a specific solution for a disruption as the costs that are not easily quantifiable and are related to the assessment, from the point of view of the passenger, of the delay time and/or cancellation of a flight. That is, we try to quantify in a monetary unit the cost that each minute of delay and flight cancellation has from a passenger's point of view. We propose a method that calculates this value according to the on board passenger profiles. In our opinion, this method is generic enough to embrace the different methods that airlines might use to calculate this important cost component.

- **An implementation of an *Advanced Prototype* that includes the above scientific contributions (Chapter 7 and Appendix A)**.
  We have implemented a system called *MASDIMA - Multi-Agent System for Disruption Management*, that includes the main contributions referred in this section and that uses real data from TAP Portugal. This system was used to perform the experiments according to what we describe in Chapter 8. The system was implemented with Java[10] and JADE[11] (Bellifemine *et al.*, 2004) as the agent framework.
- **A *System Classification* to classify disruption management systems (Section 3.3)**.
  We propose a classification for the systems or tools that are able or that can be used to deal with disruptions on the AOCCs. The idea is to classify not only available commercial software, but also, tools or systems proposed by researchers. We have used this classification to classify the related work we found regarding disruption management.

1.5 Thesis Structure

The rest of this thesis is divided into four parts.

Part I - Preliminaries: Provides the reader with background information regarding the main technologies used in this thesis as well as a perspective on the research carried out by the community regarding disruption management in airline operations and related areas. It also provides information regarding the airline operations control problem. This part is sub-divided into three chapters.

- **Chapter 2 - Background**: Provides some background information regarding the main paradigms, methods, technologies and algorithms used. We start by the multi-agent system paradigm and then we give some information regarding reinforcement learning and the three problem solving algorithms we use on our proposed system. This information allows the reader to better understand the contents and choices we have made regarding the distributed system we propose in Chapter 7, as well as the decision mechanism we propose in Chapter 6. We also provide some information about the structure of organizations, that will allow the reader to better understand Chapter 5, where we propose an improved agent-oriented methodology.
- **Chapter 3 - State of the Art**: Provides a perspective on research carried out by the community in our field of study. We start by proposing a classification for the systems we have studied when reading the literature regarding disruption management in airline operations. An extensive start-of-the-art regarding disruption management in airline operations is presented and we classify those works using the classification system proposed by us. For ease of reading we have included tables that summarize the most relevant information of the comparative analysis that was made. We include a summarized start-of-the-art regarding automated negotiation, that allows the reader to better understand the negotiation protocol we propose in Chapter 6. Likewise, a summarized state-of-the-art regarding agent-oriented software engineering is included at the end of the chapter. This allows the reader to better understand Chapter 5, where we proposed an improved agent-oriented methodology.

[10] http://www.java.com
[11] http://jade.tilab.com/

- **Chapter 4 - The Airline Operations Control Problem**: Provides detailed information regarding the airline operations control problem (AOCP), i.e., the main object of study of this thesis. We start by introducing the airline scheduling problem (ASP), the one that precedes the AOCP. Then, we give details about the airline operations control center (AOCC) organization and information sources, and we describe the typical problems affecting the airline operations as well as the current disruption management process adopted by the airlines and the costs involved. This chapter is of utmost importance for the reader to understand the problem we want to solve and, as such, to understand the design choices that are behind what we propose in Chapter 6 and Chapter 7.

Part II - The Main Thing: It is the core of this thesis. It includes the research contributions of our work, trying at the same time, to prove the hypotheses presented in Section 1.3.2. This part is sub-divided into four chapters.

- **Chapter 5 - Porto - An Improvement to Gaia**: Proposes an improvement to the *GAIA* methodology called *PORTO*, which is one of the main contributions of this thesis, and results from applying it to analyze, design and develop MASDIMA - Multi-Agent System for Disruption Management (Chapter 7 and Appendix A). Each phase of the methodology is presented, using as an example the MASDIMA system. This will allow the reader to better understand the concepts together with the proposed methodology and, at the same time, get acquainted with the system we have developed as a proof of concept for our approach to the disruption management problem.
- **Chapter 6 - Generic Q-Negotiation Protocol**: Provides a formal definition of a negotiation protocol with adaptive characteristics, which is one of the main contributions of this thesis and is used in the MASDIMA system as a decision mechanism. We also present two application examples in different application domains to show that the protocol is generic enough to be used in such heterogeneous environments. This chapter is our contribution to the *second hypothesis*.
- **Chapter 7 - A New Approach for Disruption Management in AOCC**: Presents our new approach for disruption management in the airline domain, including how we represent the AOCC using a Multi-Agent System (MAS), a computational organization of intelligent agents. Besides providing new or updated processes and mechanisms for the AOCC we also aim to focus on the AOCC as an entity, trying to change the way AOCCs are setup and organized and not only to provide DSS[12] tools. Our main focus in this chapter is on the conceptual aspects of our approach. This chapter is our contribution to the *first*, *third* and *fourth* hypothesis.
- **Chapter 8 - Experiments**: Presents the experiments we have performed regarding the hypotheses presented in Section 1.3.2. We start by describing the experimentation scenarios under analysis as well as the metrics used. The approaches or methods we have used to perform the experiments are also presented as well as the results. The chapter ends with a discussion regarding the results obtained from the experiments.

Part III - Conclusions: Presents the conclusion of our work. This part has only one chapter.

- **Chapter 9 - Conclusions**: Provides the results regarding the hypotheses presented in Section 1.3.2 and a critical appreciation regarding the main contributions presented in Section 1.4.

[12] Decision Support Systems

This chapter also identifies limitations and future work and ends with a brief remark regarding the application of our work in real world companies.

Part IV - Appendixes: Includes additional information that, although important, makes more sense to be used as a reference.

- **Appendix A - MASDIMA - Multi-Agent System for Disruption Management**: Provides a description of the features of the advanced prototype we have developed to test our ideas.
- **Appendix B - MASDIMA - Code Examples and Diagrams**: Provides some examples of the JAVA code used to implement some features and a full sequence diagram of the GQN protocol we propose in Chapter 6.
- **Appendix C - MASDIMA - Costs Information**: Provides the information used to calculate the costs regarding the three dimensions of the problem: aircraft, crew and passengers.
- **Appendix D - Delay Codes**: Provides the IATA delay code tables as well as the IATA and TAP delay code classification, that support the information provided in Chapter 4.

Part I
Preliminaries

The goal of **Part I - Preliminaries** is mainly to provide the reader with background information regarding the main technologies used in this thesis as well as a perspective on the research carried out by the community regarding disruption management in airline operations and related areas. It also provides information regarding the airline operations control problem. Calling it preliminary does not mean that we do not present important information. It is structured as follows.

In **Chapter 2 - Background**, we provide some background information regarding the main paradigms, methods, technologies and algorithms used. We start by the multi-agent system paradigm and then, we give some information regarding reinforcement learning and the three problem solving algorithms we use on our proposed system. This information allows the reader to better understand the contents and choices we have made regarding the distributed system we propose in Chapter 7 as well as the decision mechanism we propose in Chapter 6. We also provide some information about the structure of organizations, that allows the reader to better understand Chapter 5, where we proposed an improved agent-oriented methodology.

In **Chapter 3 - State of the Art**, we provide a perspective on the research carried out by the community in our field of study. We start by proposing a classification for the systems we have studied when reading the literature regarding disruption management in airline operations. An extensive state-of-the-art regarding disruption management in airline operations is presented and we classify those works using a classification system proposed by us. For ease of reading we have included tables that summarize the most relevant information of the comparative analysis that was made. We include a summarized state-of-the-art regarding automated negotiation, that allows the reader to better understand the negotiation protocol we proposed in Chapter 6. Likewise, a summarized state-of-the-art regarding agent-oriented software engineering is included at the end of the chapter. This will allow the reader to better understand Chapter 5, where we proposed an improved agent-oriented methodology.

In **Chapter 4 - The Airline Operations Control Problem**, we provide detailed information regarding the airline operations control problem (AOCP), i.e., the main object of study of this thesis. We start by introducing the airline scheduling problem (ASP), the one that precedes the AOCP. Then, we give details about the airline operations control center (AOCC) organization and information sources, and we describe the typical problems affecting airline operations as well as the current disruption management process adopted by the airlines and the costs involved. This chapter is of the utmost importance for the reader to understand the problem we want to solve and, as such, to understand the design choices that are behind want we propose in Chapter 6 and Chapter 7.

Chapter 2
Background

Abstract Chapter 1 introduced the main problem we address in this thesis, including the motivation, research methodology (observations, hypotheses and expected results) and the main contributions of our work. In this chapter we provide some background information regarding the paradigms, concepts and algorithms we use in this thesis, namely, *Multi-Agent Systems*, *Problem Solving Algorithms*, *Learning* and *Structure of Organizations*. It is not our intention to provide a full description of such subjects but only the necessary information for the reader to understand these scientific topics and the rationale that made us use them.

2.1 Introduction

To perform our work we use paradigms, concepts and techniques borrowed from several fields of knowledge, such as, *Software Engineering*, *Artificial Intelligence*, *Economics* and *Computer Science*. Namely, we use ideas and concepts out of the following scientific topics: *Automated Negotiation*, *Agent-Oriented Software Engineering*, *Multi-Agent Systems (MAS)*, *Problem Solving Algorithms (PSA)* and *Reinforcement Learning (RL)*.

Regarding the first two and since we provide contributions in both areas, we define the concepts and provide information regarding the work of other researchers in the next chapter, Sections 3.5 and 3.6, respectively.

In this chapter, the goal is to provide only the necessary information for the reader to understand the concepts of *MAS*, *PSA* and *RL* and the reasons for us to use them. It is not our intention to provide a detailed description of these concepts. Fortunately, there is very good literature about all of these topics. For example, the books written by (Wooldridge, 2009b), (Russell & Norvig, 2010), (Weiss, 1999) and (Sutton & Barto, 1998) will provide all the well established and detailed information necessary.

In our opinion, *MAS* is a powerful paradigm to represent organizations, their individuals and the surrounding environment. Considering the study we have performed, the observations made (Section 1.3.1) and the personal knowledge we have about how the Airline Operations Control Centers (AOCC) are organized, we found it to be a natural metaphor for it. Additionally, as we stated in Chapter 1, it has characteristics (Wooldridge, 2009a; Elamy, 2005) that will help to propose a *Distributed* and *Decentralized* approach to *Integrated* and *Dynamic* disruption management in airline operations control, and this fact helps to achieve the objective of our *first hypothesis* (Section 1.3.2).

To achieve the goal of our *third hypothesis* we need to implement algorithms that allow to find candidate-solutions according to the expertise of the several roles found in the AOCC. There are several problem solving algorithms that can be used. We found two of them, *Simulated Annealing* (Kirkpatrick *et al.*, 1983) and *Shortest-Path* (Dijkstra, 1959), to be the right ones to be used at this moment. Nevertheless, our approach allows many others to be used.

Finally and according to our *second* hypothesis, a learning mechanism will help to find better solutions in an environment that changes very often. From the study we have done, we believe that reinforcement based learning mechanisms have the necessary characteristics to achieve good results, and that is the reason why we chose them.

The rest of this chapter is as follows. In Section 2.2 we provide some information about *MAS* and in Section 2.3 about two of the problem solving algorithms we use in our work. In Section 2.4 we give information regarding Learning and the reasons why we chose *RL* and in Section 2.5 we provide information about the structure of the organizations, that will help the reader to understand some of the decisions we made regarding the *MAS* we conceptualized. We end this chapter with a summary in Section 2.6.

2.2 Multi-Agent Systems

A Multi-Agent System[1] (MAS) is, typically, a software system composed of multiple interacting intelligent agents (simple *agents* from now on) within an environment. The environment can be both computational and physical and it can be open or closed, i.e., an environment where the agents can enter and leave freely, or not, respectively.

In the case of computational environments and according to the behavior of the agents, the environment can be competitive (e.g., Electronic Commerce (Lomuscio *et al.*, 2000; Malucelli *et al.*, 2006) or Virtual Enterprise automatic formation (Rocha & Oliveira, 1999)), cooperative (e.g., Crisis Management (Hemaissia-Jeannin *et al.*, 2008), Train Dispatching (D'Ariano & Hermelrijk, 2006) and Air Traffic Management (Raffard *et al.*, 2005; Wollkind *et al.*, 2004)) or simultaneously have both competitive and cooperative characteristics (e.g., Supply Chain Management (Adacher *et al.*, 2008)).

The agents can be physical-like robots or humans or even other computer-based systems, but, in these cases, they are typically *agentified* (Zambonelli *et al.*, 2003), i.e., a software agent is built that will allow the physical entity to interact with the other agents in the MAS.

There are different types of agents. For example, and considering their behavior, the agents can be *reactive* (perceive the environment and react to some change in it), *pro-active* (exhibiting goal-direct behavior by taking the initiative to satisfy their design goals) or *social* (interact with other agents in order to satisfy their goals). Some of them might also express *emotional* behavior (Pereira *et al.*, 2008, 2005).

Perhaps the most consensual property of agents is *autonomy*, which we can define as *the capacity of the agent to have autonomous action in the environment in order to meet its objectives* (adapted by us from (Wooldridge, 2009c)). Beyond this characteristic there is little agreement and even regarding *autonomy* it might mean different things for different people. In Chapter 7, Section 7.2 we explain what characteristics of agents and *MAS* are important for us to adopt this paradigm.

In a MAS, for the agents to interact it is necessary to have an infrastructure that specifies a *Communication Language*, a *Domain Language* and an *Interaction Protocol*.

The objective of a *Communication Language* is to facilitate the communication in agent's interactions. Two major proposals have been advanced, namely the *Knowledge Query and Manipulation Language* (Finin *et al.*, 1994) and *Foundation for Intelligent Physical Agents' Agent Communication Language* (FIPA-ACL) standard (FIPA, 2002a). We have adopted the latter as part of

[1] Most of the information provided in this section results from reading the books (Wooldridge, 2009b; Weiss, 1999)

our *Generic Q-Negotiation Protocol* (GQN), one of the main contributions of our work specified in Chapter 6.

A *Domain Language* is necessary so that the agents are able to refer and understand the concepts of the domain, proposals, time and, of course, the object of negotiation. This includes the attributes under negotiation as well as the constants that represent the negotiation attributes' value and any other needed symbols. Efforts have been made to provide standard domain languages that can be used by agents in heterogeneous environments. For example, *DARPA Agent Markup Language* (Hendler & McGuiness, 2000) and *W3C Web Ontology Language* (OWL) (McGuiness & van Harmelen, 2003).

An *Interaction Protocol* is necessary to manage the exchange of a series of messages among agents, i.e., a conversation. As stated before, one of the main contributions of our work is a protocol called GQN. There are dozens of interaction protocols for systems of agents, making it impossible to mention all of them here. However, in Chapter 3, Section 3.5 we detailed some of the existing ones used for automatic negotiation.

Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve. Intelligence may include some systematic, functional, procedural or algorithmic search, find and processing approach. Some of these characteristics are presented in our approach as it is possible to see in Chapter 7.

2.3 Problem Solving Algorithms

Problem Solving Algorithms[2] is an expression used to name any algorithm used for problem solving, ranging from the ones that achieve optimal solutions to the ones that achieve sub-optimal solutions. In this section we will briefly explain the ones we have implemented in the *Specialist agents* (Section 7.7) included in the MAS that we propose for *Disruption Management in Airline Operations Control* (Chapter 7), *Hill Climbing*, *Simulated Annealing* (Kirkpatrick *et al.*, 1983) and *Dijkstra Shortest-Path* algorithm (Dijkstra, 1959). Before proceeding, it is important to point out that the MAS we proposed *supports any other type of problem solving algorithm*. The only thing that is necessary to do, is to develop a specialist agent that implements that algorithm and add it to the system. This is one of the advantages of the MAS paradigm: modularity and scalability.

Hill Climbing is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element in the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating this until no further improvements can be found. *Hill climbing* is good for finding a local optimum (a solution that cannot be improved by considering a neighboring configuration) but it is not guaranteed to find the best possible solution (the global optimum) out of all possible solutions (the search space).

Thinking in terms of a search for a minimum in a generic domain and analogously to the physical process of controlled heating and cooling of a material, each step of the *Simulated Annealing* (figure 2.1) algorithm replaces the current solution by a random "nearby" solution, chosen with a probability that depends on the difference between the corresponding function values and a global parameter T (called the temperature), that is gradually decreased during the process. The depen-

[2] Most of the information provided in this section results from reading the books (Russell & Norvig, 2010) and (Cormen *et al.*, 2001)

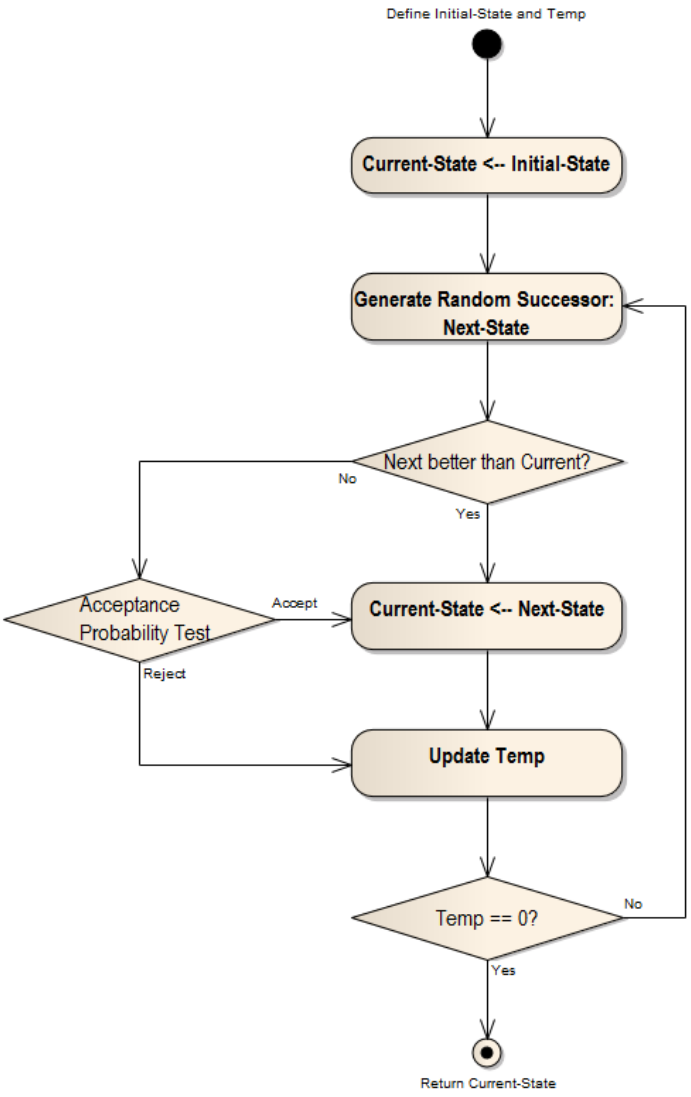


Fig. 2.1 Simulated Annealing algorithm.

dency is such that the current solution changes almost randomly when T is large, but increasingly "downhill" (improve) as T goes to zero. Allowing "uphill" (worse) moves saves the method from becoming stuck at local optima, which is a problem of other known algorithms.

Dijkstra's Shortest-Path algorithm[3] is a graph search algorithm that solves the single-source shortest path problem for a graph with nonnegative edge path costs, producing a shortest path tree. For a given source node in the graph, the algorithm finds the path with the lowest cost (being it monetary, distance, etc.) between that node and every other one.

The *Hill Climbing* and *Simulated Annealing* algorithms were used in the *Specialist Agents* to get solutions to the aircraft and crew resource problems. Sections 7.7.1 and 7.7.2, respectively, show how we have applied these algorithms to this problem.

The passenger problem dimension is different from the aircraft and crew's. It can easily be modeled as a shortest path problem, since the objective is to find the least costing path for each

[3] conceived by Dutch computer scientist Edsger Dijkstra

passenger to reach his destination airport from a starting airport. Section 7.7.3 shows how we have applied the Dijkstra's algorithm to this problem.

2.4 Learning

The goal of this section[4] is not to detail all the possible learning methods. The idea is to present the rationale behind our choice to use reinforcement learning in our proposed approach and to provide a brief information about this subject.

According to (Russell & Norvig, 2010) and (Sutton & Barto, 1998), learning in MAS can be performed by different methods, namely, *supervised* learning, *non-supervised* learning and *reinforcement* learning.

In *Supervised* learning the learner agent learns by comparing the result with the information provided by a supervisor agent, that needs to have a set of examples previously obtained. This kind of learning does not seem adequate to be used in our application domain, because, as we stated in the *fifth observation* (Section 1.3.1) *the information available might change often and unexpectedly making it very dynamic*. As such, it could lead to inefficiency in obtaining a solution. (Sutton & Barto, 1998) state that this kind of learning is not the best one to use for learning through agent interaction. Nevertheless, there are some remarkable examples of MAS that use supervised learning like the one in (Goldman & Rosenschein, 1995).

In *Non-supervised* learning, the agent tries to learn by a trial-and-error process, i.e., after the execution of an action, the agent analyzes the environment where the action was performed and reaches its own conclusions. This kind of learning is more adequate to the preprocessing of large quantity of unstructured data and it is intrinsically connected to data mining techniques.

In *Reinforcement* learning, the agent learns by experimentation. In this kind of learning, there is the concept of state, action and reward. When in a specific state s and after selecting an action a, which will be executed, the agent reaches a new state s^*. Depending on the destination state it will receive a reward r, that will penalize or award the executed action a.

Reinforcement learning includes three classes of algorithms: *Dynamic Programming*, *Monte Carlo Methods* and *Temporal Difference Methods*. We are not going to discuss here each one of these class of algorithms, since it is easily available in (Sutton & Barto, 1998). However, and as we stated previously, in the application domain of our study, the information is not static, changing often and unexpectedly. Because of that, we need a learning mechanism that is able to learn in each interaction and not only at the end of the (negotiation) process. As such, reinforcement learning and more specifically, the *Temporal Difference Methods*, is a better method to use considering these requirements.

In Chapter 7, Section 7.6.3 we detail the implementation of *Q-Learning* and *SARSA*, two *Temporal Difference Methods* of reinforcement learning that we decided to use.

[4] Most of the information provided in this section results from reading the books (Russell & Norvig, 2010) and (Sutton & Barto, 1998)

2.5 Structure of the Organizations

In this section[5] we are going to briefly introduce this topic and present the relation that, in our opinion, exists between the success of an organization (in this particular case the AOCC) and its structure. This relation together with our view of an AOCC as a distributed system, will contribute to sustain the design choices we have made regarding the multi-agent system organization structure that we propose to represent the AOCC (see chapter 7).

The structure of an organization is fundamental for the success of that organization in accomplishing its objectives. From that point of view an AOCC, considering the importance it has in the commercial success of an airline (Piques, 2006), should have a structure that is appropriate to the company's dimension, allowing the best interaction and communication between the several specialists and, at the same time, get the best possible benefit from the available information systems and/or existing decision support tools. It is possible to adopt an organizational view and see the AOCC as a distributed system composed by several sub-systems, according to the expertise and/or skills that should exist (Castro, 2007), namely: *operations controllers*, *aircraft scheduling*, *crew scheduling* and *flight dispatch*, among others. When doing that, we need to be aware of some characteristics that will help to define the best structure.

In his seminal work, Mark S. Fox (Fox, 1981) wrote that "The human mind's processing capacity is limited" and this fact results in what (March & Simon, 1993) call *Bounded Rationality*. This characteristic implies that a person has a limit regarding two things: the information that he/she is able to deal with and the control detail that he/she is able to handle. When the tasks get bigger and more complex, we need to find an effective way of limiting the information increase as well as to control complexity.

Bounded Rationality is the main factor in the evolution of the organizations (with multiple persons) from an unregimented group to more structured alternatives. Considering that a computer processor also has a limited processing capacity it is possible to extend this metaphor to programmed computer systems, whether distributed or centralized (Fox, 1981).

Continuing to see the AOCC as a distributed system, there are two main characteristics that organizations must have that we also need to consider: a *structure* and a *control regime*. Table 2.1 shows the possible organization structure according to (Fox, 1981).

In table 2.2 we can see the several control regime types according to (Zambonelli *et al.*, 2003). The structure and control regime alone are not sufficient to determine how a system should be distributed. Mark S. Fox (Fox, 1981) refers that the most important features to consider are:

1. *Task Complexity*: Most of the tasks are related to re-scheduling of one or more resources (aircraft and/or crew members). This has to be done in a short period of time, reducing the negative impact on passengers and, at the same time, take into account that the re-scheduling might propagate to other flights. All these tasks are very complex.
2. *Uncertainty*: It is difficult to forecast when and what events can lead to changes on daily airline operations. From meteorologic conditions to aircraft malfunctions and/or crew members that do not report for duty, there are several possible events that may affect the operation.
3. *Resource Restrictions*: The main type of resources in an airline are the aircraft and the crew members. However, sometimes, the resolution of unexpected events is not dependent only on those two types of resources. Maintenance facilities, *slots* for overfly and/or landing a specific

[5] Most of the material found in this section was presented in the monograph *Centros de Controlo Operacional - Organização e Ferramentas* (Castro, 2008) (in Portuguese).

Table 2.1 Organizations Structure

Structure	Characteristics
One Person	Performs all tasks. Works while available resources exist to achieve the goal. It is not useful when the tasks' complexity and/or work increases.
Group	Allows cooperative coordination of the individuals to achieve a common goal. As soon as the group increases (in terms of members), the cost to make a collective decision also increases. From that moment this kind of structure is less useful.
Simple Hierarchy	It has two levels: the top level contains a decision maker that coordinates the efforts of people in the low level. This structure is not useful from the moment when the only decision maker is unable to process all information (*Bounded Rationality*).
Uniform Hierarchy	There are several levels of management to ensure that decision making is adequate and centralized. Each level of the hierarchy acts as a for the information and for the decisions that are propagated upwards. As the hierarchy grows in size and number of products, the competition for the same resources starts. The allocation of resources starts to get more complex.
Multidivisional Hierarchy	The organization is divided into product lines or services. Each division has full control of the production strategy. As the organization grows, so does the problem of coordination and the amount of information to be processed.
Price or Market System	In this system, all forms of control between the divisions are eliminated. All communication is contained in a purchase contract of a product or service. The control is now exercised by the price of the product or service.
Collective Organization	The hierarchy is divided into separate organizations that cooperate to achieve a common goal. From a certain point of view it may be seen as a set of organizations that share long-term contracts.

Table 2.2 Control Regime Types

Control Regime	Characteristics
work Partition	Each element has the same function and provides the same type of service/activity.
Work Specialization	Each element provides a service/activity.
Market Models	The workload distribution and service delivery is carried out based on this type of model. Used when organizations are large and when selfish and competitive behavior begin to emerge.

flight region and/or destination, among others, are scarce resources that complicate seriously the task of solving the problems by the AOCC. As (Piques, 2006) says, the goal is that "(...) the resources are available in the right quantity, at the right moment and with the desired quality (...)". If we take into account all the previous factors, it is easy to see that the resources are a serious restriction to achieve this goal.

All the above features or factors exist in the AOCC.

Continuing our idea of seeing the AOCC as a distributed system and considering the organization structure, the control regime and the factors referred on previous paragraphs, the question that we should ask is: *What is the best organization that an AOCC should adopt considering the type of tasks performed*? According to (Fox, 1981) we need to address three issues to be able to make the best decision:

1. Task(s) characteristics.
2. A set of organization structures and control regimes that are able to handle the previous tasks and,
3. A way of measuring the organization performance.

The first two, task characteristics and organization structure, have already been discussed. Regarding the third, measuring the organization performance, it is important to point out that the goal of the task (performed by the organization) provides the necessary criterion. Goals like: minimize resource consumption, maximizing production, improve the quality, and so on, are easily found in the organizations. These goals are similar to the time and space reduction, increase processing capacity and produce results of greater validity, in the distributed systems.

An interesting measure technique is the *Transaction Analysis* (Williamson, 1983). In this case the word *transaction* has a wider scope than usual. It includes contracts, communication of information, monitoring, delegation and control and most activities that require interaction amongst participants in an organization or market. Since we need resources to handle transactions, we can say that transactions are too complex when they require more resources than the ones available (*Bounded Rationality*). Thus, the complexity reduction becomes a problem of minimizing the resource consumption, that is, transactions between processes should be properly structured so that resource consumption is minimized.

Another transaction characteristic is the assumed difference in the information, motivation and behavior among participants in a transaction. If we detail the transactions among the participants in an organization, it will be possible to measure the effectiveness of alternative organizational structures. Fox (Fox, 1981) presents a large variety of techniques to reduce the complexity and uncertainty as well as organizations that are appropriate to reduce its effects. The author uses the analysis of transactions as a measure of the uncertainty and complexity of organizational structures.

In general the complexity and uncertainty are two important factors to take into account when we decide how to structure an organization. The complexity pushes the task distribution leading to a more distributed structure. Uncertainty tends to force the structure in the opposite direction, integrating tasks in a more more hierarchical structure.

Answering to the question about the best organizational structure for the AOCC and considering all that we have said previously, we might reach the *preliminary conclusion that it will be a mixed hierarchical/distributed structure*.

As stated in the beginning of this section, the idea is to provide the reader with some background about the structure of the organizations, the relation that exists between the adopted structure and

the success of the organization and to sustain our view of an AOCC as a distributed system. In chapter 4, specifically in section 4.3, we will show common structures adopted currently by the AOCC and in chapter 7 we will present the rationale behind the design choices that defined the structure of our proposed MAS.

2.6 Chapter Summary

The objective of this chapter was twofold:

- Provide some brief information regarding some of the paradigms, concepts and algorithms we will use later on in our work, specifically, the *Multi-Agent System*, *Problem Solving Algorithms* and *Reinforcement Learning*.
- Provide a rationale for having adopted this paradigms, concepts and algorithms.

It was not our intention to provide a detailed description of these concepts, since they are available in very good literature such as (Wooldridge, 2009b), (Russell & Norvig, 2010), (Weiss, 1999) and (Sutton & Barto, 1998).

Additionally, we have provided an introduction to the topic of *Structure of the Organizations* and the relation that, in our opinion, exists between the success of an organization and its structure. This relation together with our view of an AOCC as a distributed system, will contribute to sustain the design choices we have made regarding the multi-agent system organization structure that we propose in Chapter 7, to represent the AOCC.

In the next chapter we will present the state of the art regarding disruption management in airline operations (including a taxonomy to classify such systems), as well as, some information regarding *agent oriented software engineering* and *automated negotiation*, two very important paradigms that are on the foundations of two of the main contributions of our work.

Chapter 3
State of the Art

Abstract In Chapter 2 we provided some background about the concepts, methods and techniques that we use in the rest of this thesis. The main goal of this chapter is to present a comprehensive analysis of the work that has been done by other researchers and developers regarding disruption management in airline operations control. To better compare the existing research work we propose a classification scheme that can be used to classify research work as well as tools or systems in use at airlines, along several relevant perspectives. For ease of reading, we have included tables that summarize the comparative analysis that was made. Additionally and since we propose in this thesis a new automated negotiation protocol and a new agent-oriented software engineering methodology, we here also mention some work related to automated negotiation and agent-oriented software engineering. It is important to point out that, regarding these last two topics, our goal is not to provide a full description of the state-of-the-art but, simply, to convey enough information for the reader to understand our proposal.

3.1 Introduction

According to (Yu & Qi, 2004), *Disruption Management* (in general) can be formally defined as:

> *At the beginning of a business cycle, an optimal or near-optimal operational plan is obtained by using certain optimization models and solutions schemes. When such an operational plan is executed, disruptions may occur from time to time caused by internal or external uncertain factors. As a result, the original operational plan may not remain optimal, or even feasible. Consequently, we need to dynamically revise the original plan and obtain a new one that reflects the constraints and objectives of the evolved environment while minimizing the negative impact of the disruption. This process is referred to as disruption management.*

As we stated in Chapter 1, our work is focused on disruption management (or operations recovery) in airline operations control, through the application of the agent and multi-agent systems (*MAS*) paradigm. However, as it is possible to see from the definition given by *Yu et. al.*, disruption management is a general topic that can be recognized as relevant on other domains, for which solutions are proposed with or without using the *MAS* paradigm.

For example, (Zhu *et al.*, 2005) study the problem of how to react when an *ongoing project* is disrupted. Their focus is on the *resource constrained scheduling problem* with finish-start precedence constraints and they use integer programming to model and solve the problem. In the *supply chain* domain (Ji & Zhu, 2008), developed a mathematical model to offer decision making support for adopting reasonable disruption management strategies. The *disrupted vehicle routing problem* is introduced by (Mu *et al.*, 2011). The goal is to define a new route after a vehicle break down. The authors apply two Tabu Search algorithms (Glover, 1989, 1990) and compare their results with those obtained by using an exact algorithm. In the *manufacturing industry domain* (Leitão, 2011) a holonic disturbance management architecture on the ADACOR[1] foundations is introduced. The authors use the *MAS* paradigm to propose a distributed solution to this problem. As a final example,

[1] A manufacturing control system proposed by the same authors

we would like to point out the work of (Morgado & Martins, 2012) regarding real-time dispatching with provisions to handle major disruptions in railroad operations. The authors present their decision support tool that is in use in several railroad operators in the North of Europe.

The *MAS* paradigm has been used in several application domains, including for other air transportation problems. To the best of our knowledge, we believe that we were the first to use this paradigm to represent the *Airline Operations Control Center (AOCC)* as an organization of agents (Castro & Oliveira, 2007; Castro, 2007). Regarding the use of agents in other domains a very brief list follows: (Jonker *et al.*, 2005) propose a multi-agent system for *Air Traffic Control (ATC)* Tower operations. In the aviation domain, but in a different context, (Tumer & Agogino, 2007) present a multi-agent system for air traffic flow management. Another use of agents in the context of collaborative traffic flow management is reported by (Wolfe *et al.*, 2007). Here, agents are used to compare routing selection strategies. As a last example and in a completely different domain, (Ouelhadj, 2003) developed an integrated dynamic scheduling system of steel production based on the multi-agent paradigm.

The examples above constitute an incomplete and very brief list regarding disruption management in other domains and regarding the use of the *MAS* paradigm, for the reader to have just an idea that the same kind of problem is being addressed for several different application domains.

As we stated in Chapter 1, our focus is on the disruption management in airline operations control, i.e., to manage unexpected events that might affect flights, causing departure or arrival delays, minimizing delays and the costs involved. Airline companies spend time and money in creating optimal operation plans that maximize the revenue. However, on the day of operation, the known but unpredictable events (bad weather, aircraft malfunctions and crew absenteeism, for example) might completely change that operational plan and, consequently, the revenue objectives behind it (Clausen *et al.*, 2010).

The most important part of this chapter presents a comprehensive analysis of the work that has been done by other researchers and developers regarding disruption management in airline operations control. To be able to better compare the existing work, we propose a classification scheme to better clarify the different issues addressed and approaches taken by the current and previous research work. For ease of reading, we have included tables that summarize the comparative analysis that was made.

The rest of this chapter is structured as follows: Section 3.2 gives definitions that are important to understand some of the concepts introduced in this chapter as well as in other following chapters of this thesis. In section 3.3 we propose a classification scheme to classify AOCC tools, systems or research work. Using this scheme and other information, in section 3.4 we compare and classify the approaches reported in the existing literature on operations recovery in AOCC, making some comparisons with our proposal. In section 3.5 we show some work related to automated negotiation, since we propose a new automated negotiation algorithm in our study and in Section 3.6, we do the same regarding agent oriented software engineering methodologies. A final summarization of the chapter is presented in section 3.7.

3.2 Definitions

In this section we present some definitions that will allow the reader to better understand the next following sections. First we start by defining concepts related to disruption and, then, those related

with the recovery process. Some definitions related with aircraft and crew roster and passenger itineraries, are also provided. Finally, we give a few definitions related with automated negotiation for the reader to better understand the information provided in Section 3.5.

3.2.1 Related to Disruption Concepts

Definition 3.1. *Disrupted Aircraft*
An aircraft which cannot complete its original schedule.

Definition 3.2. *Disrupted Airport*
The airport where the disrupted flight occurred.

Definition 3.3. *Disrupted Crew Member*
A crew member which cannot complete its original schedule.

Definition 3.4. *Disrupted Flight*
A flight that, due to an unexpected event, cannot depart and/or arrive on its schedule time.

Definition 3.5. *Disrupted Passenger*
1: A passenger that has lost one or more flight connections due to a disrupted flight.
2: A passenger whose itinerary contains a disrupted flight.

Definition 3.6. *Disruption in Airline Operations*
An interruption to the regular flow or sequence of the airline operations plan.

Definition 3.7. *Disruption Problem and Problem Dimension*
In a disruption management context a problem dimension is a subpart of the disruption problem that needs to be solved. Each problem dimension must be characterized by one or more attributes. A disruption problem is characterized by the union of its problem dimensions. In Section 6.3 a more formal definition of Problem and Dimension is given. In airline operations, a disruption problem is typically composed of three sub-parts: aircraft, crew and passenger.

3.2.2 Related to Recovery Processes

Definition 3.8. *Aircraft Recovery (AR)*
The process of assigning individual aircraft to a disrupted flight minimizing a specific objective (usually the cost and flight delay) while complying with the required rules. The output will be a set of actions related to the assignment of aircraft to flights, to be applied on the airline operational plan, so that the disruption is solved. It also corresponds to the resolution of the aircraft dimension of the disruption problem (see Definition 3.7).

Definition 3.9. *Crew Recovery (CR)*
The process of assigning individual crew members to a disrupted flight minimizing a specific objective (usually the cost and delay) while complying with the required rules. As with the previous, the output will be a set of actions related to the assignment of crew members to duty activities, to be applied on the airline operational plan, so that the disruption is solved. It also corresponds to the resolution of the crew dimension of the disruption problem (see Definition 3.7).

Definition 3.10. *Flight Recovery (FR)*
The process of repairing a flight schedule after a disruption, through specific actions like delay, cancel or divert flights from their original schedule, so that the flight delay is minimized. It is closely related to the aircraft and crew recovery process, since it depends on the availability of these resources to be successfully solved.

Definition 3.11. *Integrated Recovery (IR)*
A process that is able to recover all problem dimensions (see Definition 3.7) separately but not simultaneously. Usually, the dimensions are solved sequentially, that is, the output of one dimension is the input of another and the order by which they are solved, imposes an importance factor to the dimensions. The output will be a set of actions related to the aircraft and crew dimension to be applied on the airline operational plan and, for the passenger dimension, new itineraries for the disrupted passengers.

Definition 3.12. *Passenger Recovery (PR)*
The process of finding alternate itineraries, commencing at the disrupted passenger location and terminating at their destination or a location nearby, while minimizing a specific objective (usually the passenger trip time and the airline costs). It also corresponds to the resolution of the passenger dimension of the disruption problem (see Definition 3.7).

Definition 3.13. *Partial-Integrated Recovery (PIR)*
A process that is able to recover at least two, but not all, of the problem dimensions (see Definition 3.7), simultaneously or not.

Definition 3.14. *Simultaneously Integrated Recovery (SIR)*
A process that is able to recover all problem dimensions (see Definition 3.7) simultaneously, without imposing a degree of importance to any of the dimensions, i.e., all dimensions are of equal importance.

3.2.3 Related to Aircraft, Crew Roster and Passenger Itinerary

Definition 3.15. *Aircraft Readiness*
Time at which the aircraft is ready for another flight after arriving at an airport. This implies that all regulations and restrictions are contemplated.

Definition 3.16. *Crew member Activity*
Time span occupied. It can range from the time performing a flight to vacation time.

Definition 3.17. *Crew member Readiness*
Time at which the crew member is ready for another activity after ending one. This implies that all regulations and restrictions are contemplated.

Definition 3.18. *Passenger Trip Time*
The elapsed time between the departure time of the passenger's first flight and the arrival time of the passenger's last flight.

3.2.4 Related to Automated Negotiation

Definition 3.19. *Adaptive (Adap)*
A negotiation model or protocol is adaptive when it has agents with adaptive behavior, i.e, that use some kind of learning mechanism. For example, to learn how to formulate new proposals.

Definition 3.20. *Agents*
In a negotiation model it is the number of agents representing each side, i.e, one-to-one (1-1), one-to-many (1-M), many-to-many (M-M).

Definition 3.21. *Dimensions (Dim)*
Number of attributes of the object of negotiation, i.e, single (*S*), multidimensional with independent dimensions (*M*) or multidimensional with interdependent dimensions (*M-Interd*). In this latter case it means that the protocol or negotiation model is able to deal with the interdependencies between attributes.

Definition 3.22. *Environment (Env)*
To distinguish between a *Competitive* (*Comp.*) and *Cooperative* (*Coop.*) environment we adopted the definition given by (Hoen *et al.*, 2006), "The fundamental distinction between systems labeled as cooperative or competitive is that for the former the agents are designed to have as goal the maximization of a group utility. Competitive agents are solely focused on maximizing their own utility". To classify a system as *Semi-competitive* (*S-comp.*) we have adopted the definition given by (Luo *et al.*, 2003), "It is cooperative because agents search for a reasonable agreement for both. It is competitive because agents want to maximize its profit" and for *Semi-cooperative* (*S-coop.*) the definition given by (Zhang & Lesser, 2012), "Each agent has its own goals and maximizes its utility. However, the performance of each agent is connected to the cooperation with others and to the global performance of the system". Finally, we have defined a system as having a *Mixed* environment when it *supports agents with heterogeneous behavior, i.e., competitive, cooperative and agents with both characteristics (a single agent has competitive and cooperative characteristics). This means that they can all participate in the negotiation*.

Definition 3.23. *Knowledge (Knowl)*
In a negotiation model or protocol, it is related to the knowledge that each agent has to present a complete proposal in the negotiation (by complete we mean to propose values to all attributes that are part of the object of negotiation). If the agent is able to do that alone then it has *FULL* knowledge. Otherwise, it has *PARTIAL* knowledge and needs the collaboration of other agents to complete the proposal (e.g, engaging in a negotiating with others).

Definition 3.24. *Participation (Part)*
In a negotiation model it is the number of distinct parties (sides) participating in a negotiation, i.e, bilateral or multilateral.

3.3 A System Classification

In this section we propose a classification for the systems or tools that are able or that can be used, to deal with disruptions on the AOCCs. The idea is to classify not only commercial software available but, also, tools or systems proposed by researchers and only available at labs' environment.

In a previous work (Castro, 2008) we have classified the tools (or systems that provide those tools) in use at AOCCs in one of three categories. Now, we extend that classification to include a fourth category, so that we are able to classify some research work presented on the next section. The proposed classification scheme is as follows:

1. Models and Algorithms (MALG)
2. Database Query Systems (DBQS)
3. Decision Support Systems (DSS)
4. Automatic or Semi-Automatic Systems (ASAS)

The *MALG - Models and Algorithms* category refers to research work that proposes models (mathematical or others) and algorithms to solve any or all of the main type of problems found in AOCC, namely, aircraft/flight problems, crew problems and passenger problems. It may include some software as proof-of-concept and/or case studies and, in some cases, it is subject to experiments with real data. However, the models and algorithms were not included in any larger tool or system (although some of them were designed for that purpose).

The *DBQS - Database Query Systems* category (the most common at airlines) allows the AOCC human operators to perform queries on the existing databases to monitor the airline operation and to obtain other data essential for decision-making. For example, to obtain the expected schedule of aircraft and/or crew members, aircraft maintenance schedule, passenger booking, etc. These systems are useful and relatively easy to implement and/or acquire but they have some important disadvantages, for example, to find the best solution and to make the best decision is completely dependent on the human operator. As we have explained in (Castro, 2008) there are two problems when airline companies use only this type of systems:

1. The solution quality is dependent on knowledge and experience of the human operator and,
2. Due to the usual difficulty of the human being in leading with large volumes of data simultaneously, they do not use all the necessary information (variables) to make the best decision.

The *DSS - Decision Support Systems* category, besides having the same characteristics of the DBQS, also includes additional functionalities to support the human operators on the decision-making. For example, after a request made by a human operator, these systems are able to recommend the best solution to solve a problem related to a delayed aircraft. Some of them may just recommend a flight re-scheduling but others are able to justify the candidate solution as well as to present the solution cost. DSS systems eliminate some of the disadvantages of the DBQS systems. Namely, they are able to analyze large volumes of data and, because of that, propose solutions that take into consideration more information (variables). The decision-making still is on the human operator side but, now, he is able to make more informed and better decisions. Unfortunately, one of the biggest problems at airlines is the absence and/or the complexity of the computerized information system with all the airline operation information. This information is fundamental for the decision support tools to find the best solution for a problem. This problem, referred in (Kohl *et al.*, 2004) as the *Data Quality and System Accessibility* problem, becomes more important when we try to develop decision support systems and/or automatic or semi-automatic systems (the next category).

The goal of the fourth category of systems, *ASAS - Automatic or Semi-Automatic Systems*, is to automate as much as possible the AOCC, replacing the functional part by computerized programs. Specifically, these systems try to automate the repetitive tasks and also the tasks related to searching for the best solution (problem solving). In a totally automatic system, decision-making is also

made by the system. In a semi-automatic system, the final decision is made by the human operator. In ASAS type of systems, the AOCC does not need as much human operators as in the previous ones, to operate correctly. Usually, roles or functions related with operation monitoring, searching for solutions related with aircraft, crew or passenger problems and re-allocation of resources, are performed by specialist agents (Castro & Oliveira, 2007) replacing the human specialists. The final decision regarding the application of the solution found by these systems on the environment (for example, making the necessary changes on the airline operational plan database) depends on the human supervisor. According to (Wooldridge, 2009a) and (Castro, 2007) the agent and multi-agent systems paradigm is more appropriate to be used in this domain than any other paradigm.

Having defined these categories we will classify the research work in the next section accordingly.

3.4 Disruption Management in Airline Operations

In this section, we present the analysis we have performed to sixty research works related to this subject, published from 1984 to 2012. We do not claim that these are all the works published during this period of time. However, from our own research (Castro & Oliveira, 2009) and from (Clausen *et al.*, 2010), we believe these are the most significant ones.

Table 3.1 presents a descendant chronological order of research regarding airline disruption management, including a summary of the *main strategies and objectives* and the *main model or solver algorithm* used. We also classify each work according to the recovery type, i.e.,

- *AR - Aircraft recovery*: when a research work only solves this part of the problem (Definition 3.8).
- *CR - Crew recovery*: when a research work only solves this part of the problem (Definition 3.9).
- *PIR - Partial-Integrated recovery*: when a research work solves at least two but not all parts of the problem (Definition 3.13).
- *IR - Integrated recovery*: when a research work solves all parts of the problem separately, i.e., not simultaneously (Definition 3.11).
- *SIR - Simultaneously-Integrated recovery*: when a research work solves all parts of the problem simultaneously (Definition 3.14).

Additionally, a classification according to the system classification proposed in section 3.3 is also included in Table 3.1.

Historically, most of the work in disruption management in airline operations (also known as operations recovery) has been done using operation research methods (*OR*). For the interested reader (Barnhart *et al.*, 2003) gives an overview of *OR* air transport applications. The study we have performed, corroborates this fact: 44 out of 60 (73.3%) use typical *OR* algorithms or models. *Integer Optimization* is the most popular with 21.6% followed by *Network Flow* models and algorithms with 20%. *Column Generation (CG)* methods are also popular with 13.3% of the works using it. The rest of the research work uses metaheuristics with *local search heuristics* (15%) as the one that is used most. Two of the works use *Genetic Algorithms (GA)* and *Ant Colony Optimization (ACO)*. To the best of our knowledge, our proposal is the first one to use the agent and multi-agent paradigm.

Looking to the research from the objectives point of view, we can see that 70% of them minimize the *delays*, *cancellations* and *operational cost* and that only 7% and 3% of them, consider the

impact on passenger and *aircraft maintenance* restrictions. Some of the works (8%) prefer to maximize the *profit*, i.e., the revenue minus the operational costs, instead of the *operational costs* and 12% also include in the objectives the *total or partial cost of assignment*, i.e., the cost of allocating specific resources to activities such as a flight. Our proposal minimizes the delays and operational costs associated to the aircraft, crew members and passengers, including assignment costs related to the first two, amongst other properties.

Considering that research on this field exists since 1984, we were expecting to see more of the research proposals included in commercial tools or in tools developed in house by the airline companies. However, looking at Figure 3.1, we can see that most of them (75%) are classified by us as *MALG (Models and Algorithms)* (see Section 3.3) meaning that they were not included in tools or systems[2]. Approximately, 22% are considered *DSS (Decision Support Systems)* meaning that they were included in tools that are used by the human operators in the AOCC and only approximately 3% of them are classified as *ASAS (Automatic or Semi-Automatic Systems)*, corresponding to our proposal. Using our personal knowledge and experience of the airline operations control domain we believe that the majority of the airline companies still use tools or systems classified by us as *DBQS (Database Query Systems)*, as it is the case of TAP Portugal.

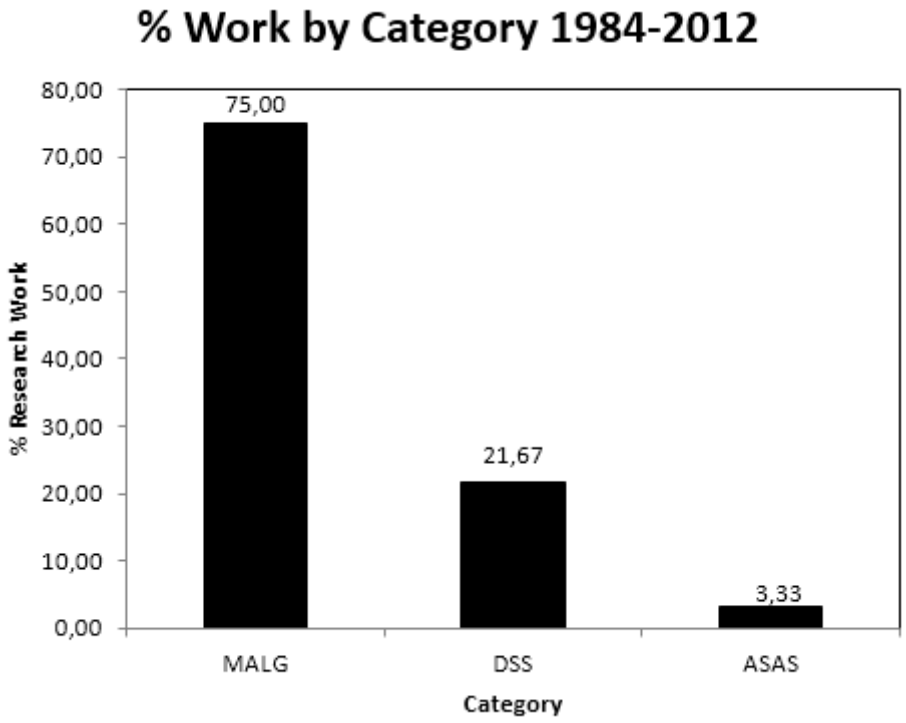


Fig. 3.1 Related work by Category from 1984-2012

If we look to the related work considering the recovery process, we can see that the majority only recovers the aircraft part of the problem followed by the ones that recover only the crew part. However, things are changing. Looking at Figure 3.2 we can see that proposals that include only the aircraft (*AR*) or the crew (*CR*) part of the problem, reach their peak in the period of 1996 to

[2] We reached at this conclusion from the information provided in the research papers by the authors. We did not conduct additional research in the airline companies to validate or not this information.

2001 and 2002 to 2007, respectively, and are now decreasing significantly. On the other hand, the proposals that include at least two but not all parts, i.e., *partial-integrated recovery (PIR)*, are increasing. In the 2008 to 2013 period there are twice as much proposals as in the 2002 to 2007 period. Regarding *integrated recovery (IR)* and *simultaneously-integrated recovery (SIR)*, there are still very few proposals and, regarding *SIR*, ours is the only one we know to the best of our knowledge.

Considering all this information, we believe that research in this domain will continue to shift to *PIR* and that, in the period from 2014 to 2019 we will start to have an increasing on *IR* and *SIR* proposals, hopefully, conducting to systems that are automatic or semi-automatic (*ASAS*).

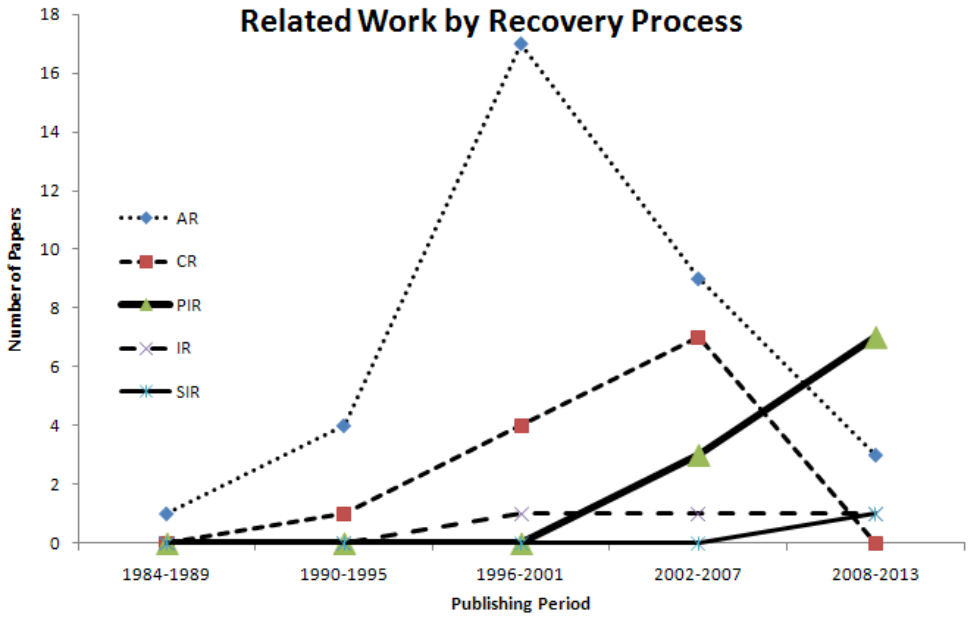


Fig. 3.2 Related Work by Recovery Process from 1984-2012

3.4.1 Literature on Aircraft Recovery

One of the first studies about the aircraft recovery problem was presented in 1984 by (Teodorovic & Guberinic, 1984). Here, the goal was to minimize the total passenger delays on an airline network when one or more aircraft are unavailable. It considers only one fleet and ignores aircraft maintenance constraints. The problem of determining a new routing and scheduling plan for the airline fleet is solved by branch and bound methods. The efficiency of the model is illustrated with a very simple example of only eight flights. This work is later extended by (Teodorovic & Stojkovic, 1990) by minimizing the total number of canceled flights and the total passenger delays. They developed a heuristic algorithm to solve this optimization problem and tested it in a small example with 14 aircraft and 80 flights. Later, (Teodorovic & Stojkovic, 1995) further extended it by including crew considerations. They developed a heuristic model, using the first in, first out (FIFO) principle and a sequential approach based on dynamic programming. This method was tested on 240 different randomly generated numerical examples. In the next sub-sections we will

detail the rest of the aircraft recovery literature, grouping them by the methods used, i.e., *OR* and Metaheuristics. A sub-section for papers related to special cases is also included.

3.4.1.1 Based on *OR* methods

Several aircraft recovery approaches are based on *network flow* models and algorithms from *graph theory*. In (Jarrah *et al.*, 1993) the authors present two network flow models: one for cancellation and one for re-timing, and this is the main disadvantage of their approach, since a trade-off between the two methods is not allowed in a single decision process. On the positive side, their method was successfully implemented in a decision support system named *SOA (System Operations Advisor)* at United Airlines. A report on this implementation and operational use is found in (Rakshit *et al.*, 1996). Here, the authors claim that *SOA* has saved more than 27,000 minutes of potential delays, which translates to $540,000 savings in delay costs, and the number of flight delays charged to aircraft controllers in systems operations control has dropped by 50 percent since the date of its implementation (1992). The work of (Cao & Kanafani, 1997a) and (Cao & Kanafani, 1997b), are basically extensions of Jarrah and Rakshit's work. They present a quadratic zero-one programming model that allows a trade-off between cancellations and delays, trying to maximize the profit and taking into consideration different cost penalties of delay and flight cancellation. The model also considers swapping different types of aircraft and takes into consideration the issues of *ferrying* (fly an aircraft without passengers to an airport to cover open flights from that airport).

The paper (Mathaisel, 1996) describes the design and implementation of a decision support system for airline disruption management. It is a system based on a network of UNIX workstations, one of them acting as a server. It integrates real-time flight tracking, aircraft routing, maintenance, crew management, gate assignment and flight planning with dynamic aircraft re-scheduling and fleet re-routing algorithms (based on network flow) for irregular operations. The model is capable of using cancellation as well as re-timing, however, the paper does not mention multiple types of aircraft or crew considerations.

The last two works that use network flow models and algorithms are (Yan & Dah-Hwei, 1996) and (Yan & ping Tu, 1997). In the former, the authors developed several *perturbed* network models for scheduling after a disruption. These network models are formulated as pure network flow problems or network flow problems with side constraints and solved using the network simplex method and Lagrangian relaxation with subgradient methods, respectively. The goal is to maximize the profit and the model also considers ferrying of aircraft. A case study on the operations of a Taiwan airline is presented. The latter is an extension of the former, that includes the possibility of dealing with multi-fleet and multi-stop flights and formulates the problem as a multiple commodity network flow problem.

A very popular method to solve the aircraft recovery problem is *integer programming (IP)* although with different network representations for the *IP* models. (Thengvalla *et al.*, 2000) uses a network flow model with side constraints that not only includes options for delaying and canceling flights but also incorporates a measure of deviation from the original aircraft routings. The model is flexible, allowing user preferences in its objective, and thereby reflecting the immediate concerns of the decision-maker in the recovery schedule. The model can be tailored by airline operations personnel to emphasize differing solution characteristics. However, it does not include crew and passenger considerations as well as aircraft maintenance requirements. The approach was tested with real data from Continental Airlines and results show that, optimal or near-optimal so-

lutions, are routinely obtained from the LP (linear programming) relaxation of the network. When integrality is not achieved, a heuristic is used that provides near-optimal feasible solutions. This work is extended by (Thengvall *et al.*, 2001) and (Thengvall *et al.*, 2003), allowing the model to deal with multiple fleets and the closure of a hub. These two works introduce three mixed-integer programming models, two of them are based on time-line networks and the other on a time-band network.

A time-band optimization model for reconstructing aircraft routings after a delay, is presented by (Bard *et al.*, 2001). The resulting formulation is an integral minimum cost network flow problem with side constraints that ensures each flight is either canceled or flown by a unique aircraft. The goal is to minimize delay and cancellation costs. The approach was tested using real data from Continental Airlines and the empirical results demonstrate that the solutions are either probably optimal or very near.

In Andersson's thesis (Andersson, 2001), the author presents a decision support tool that can solve the *flight perturbation* problem. Based on a connection network, a mixed integer multi-commodity flow model with side constraints is developed. To resolve the perturbations, the model uses flight cancellations and delays as well as multi-fleet aircraft swaps. The model also assures that the schedule returns to normal within a certain time. The goal is to maximize the revenue. Six different solution strategies are used to solve the model. One based on a Lagrangian relaxation of the mixed integer multi-commodity flow model, four based on Dantzig-Wolfe decomposition and the last based on tabu search metaheuristic. Computational tests with real problem data show that the Dantzig-Wolfe based strategies and tabu search are very promising. According to the authors, tabu search could be used in a real problem application that could provide airlines with solutions to complex perturbation problems.

In (Eggenberg *et al.*, 2007), the authors consider an heterogeneous fleet of aircraft, with regular and reserve airplanes, where the aircraft maintenance constraints are considered. The master problem is modeled as a generalized set partitioning problem and a column generation scheme is proposed to solve it. The subproblem is modeled as a constrained resource shortest path problem. An independent recovery time-band network for each aircraft is constructed making it easy to incorporate maintenance restrictions. The computational results were obtained using real world data instances.

(Wu & Le, 2012) model the aircraft recovery problem as a time-space network and developed a so-called *iterative tree growing with node combination method* to solve it. The goal is to minimize the total cost of assignment plus cancellations and delays. Aircraft maintenance and other regulations are considered. The approach was tested with data from Chinese airlines and, according to the authors, the results show that, on average, for a medium-size airline recovery, a feasible solution is found twice as fast as an exact algorithm.

Another very popular approach is to formulate the problem as a *set partitioning model*. In (Clarke, 1997) and (Clarke, 1998) the authors propose a *column generation* model based on a generalized set partitioning model with several constraints to consider aircraft maintenance, crew availability and slot allocation requirements. The objective is to maximize the profit and it considers costs related to reassignment of flights, operating costs and predetermined passenger revenue spill costs[3]. The authors propose a tree-search heuristic and a set packing based optimal solution method. The case studies presented use multiple aircraft types.

[3] Spill cost on a flight is the revenue lost when the assigned aircraft for that flight cannot accommodate every passenger.

(Rosenberger *et al.*, 2003) presents an optimization model that reschedules legs and reroutes aircraft by minimizing an objective function involving rerouting and cancellation costs. The problem is formulated as a set partitioning model with additional time slot and airport capacity constraints. The authors developed a heuristic called *Aircraft Selection Heuristic (ASR)* to select for each disrupted aircraft, a number of non-disrupted aircraft that could be used to swap with the disrupted one. The model is solved with CPLEX 6.0[4]. The proof of concept is provided using SimAir (Rosenberger *et al.*, 2000) a simulator of airline operations.

In (Andersson & Varbrand, 2004) the authors use a mixed integer multi-commodity flow model with side constraints further reformulated into a set packing model with *generalized upper bound (GUB) constraints* and using the Dantzig Wolfe decomposition. Disruptions are solved using cancellations, delays and aircraft swaps and the model ensures that the schedule returns to normal within a certain time. The model is tested on real problem data obtained from a Swedish domestic airline and the results show that it is capable of presenting high quality solutions in a few seconds and therefore can be used as a dynamic decision support tool by the airlines.

3.4.1.2 Based on Metaheuristics

The research work presented so far, regarding aircraft recovery, were based on *OR* methods corresponding to 67% of the studies we have selected. From the moment this field become more popular, contributions that use metaheuristics started to appear. In aircraft recovery we have selected 7 papers, corresponding to 21%, that use this kind of approach. The first one is from (Arguello *et al.*, 1997). Here, the authors present a *greedy randomized adaptive search procedure (GRASP)* (Feo & Resende, 1989) which enables the reconstruction of aircraft routings affected by disruptions. The goal is to minimize the costs of reassigning the aircraft considering the flight delays and cancellations. Aircraft maintenance restrictions are not considered and the method only works for a single feet of aircraft. In the proposed procedure, the neighbors of an incumbent solution are generated by flight route augmentation, partial route exchange, and simple circuit cancellation. After being evaluated, the most desirable ones are placed on a restricted candidate list and one is selected randomly, becoming the incumbent. The heuristic is polynomial with respect to the number of flights and aircraft. The procedure was tested using real data from Continental Airlines B757 fleet, with 16 aircraft and 42 flights.

Methods based on *local search* are presented in (Løve *et al.*, 2005). The authors describe and implement a heuristic which is able to generate feasible revised flight schedules of good quality in less than 10 seconds. The heuristic considers delays, cancellations and reassignments simultaneously and balances the trade-off between these options, through weights incorporated in the objective function. The main goal is to maximize the profit. The heuristic is based on a network formulation where nodes are either aircraft or flights. Based on this representation the existing solution is altered by swaps that exchange flights between two aircraft. In this paper the data used to test the approach was randomly generated. However, a feasibility study of the solutions provided by this approach was conducted using real data from British Airways during the DESCARTES project, as described in (Kohl *et al.*, 2004).

Two metaheuristics, *Tabu Search* (Glover, 1989, 1990) and *Simulated Annealing* (Kirkpatrick *et al.*, 1983), are used and compared in the work of (Andersson, 2006). The heuristics uses a

[4] CPLEX solver from IBM is a high performance solver for linear programming (LP), mixed integer programming (MIP) and some quadratic programming (QP/QCP/MIQP/MIQCP) problems.

tree-search algorithm to find new schedules for the aircraft, that allows flights to be delayed or canceled and aircraft to be swapped. Both heuristics present good results, however, tabu search stands out as the preferable solution strategy, both when considering the quality of the solutions and the robustness of the method. Within 15 seconds, the tabu search always produces a solution that is less than 0.3% from the best known solution. The tool developed in the work is meant for assisting the AOCCs users. When a disruption appears a ranked set of structurally different solutions is presented to the user. The user can select one of the solutions in the list, or choose new weights, i.e. new objective function coefficients, and solve the problem again. Tests were carried out using data from a Swedish domestic airline.

A *Multi-Objective Genetic Algorithm* is used by (Liu *et al.*, 2006) to construct new feasible aircraft routings. It only considers a single fleet and the options are to swap flights between aircraft or to delay flights. It has three objectives: minimize delay costs, swap costs and aircraft assignment costs. The genetic algorithm chromosome represents the allocation of flights to a specific aircraft. The approach was tested with real data from a Taiwan domestic airline consisting of 7 aircraft and 70 flights. This approach was later extend in (Liu *et al.*, 2008) by adding multi-fleet support and additional objectives such as, minimize turn-around times, flight connections and swaps, cancellations and total flight delay.

An approach that integrates *Tabu Search* with classical optimization methods is presented in (Yang, 2007). The objective is to minimize the schedule recovery costs associated with flight schedule modifications and deviations from the original route, i.e, delay, cancellation and aircraft swap costs. The approach involves a stand-alone tabu search that minimizes the sum of the cost of delays, cancellations and swaps and an hybrid method which combines a time-space network flow model with side constraints and a limited tabu search. According to the authors, their research shows conclusively that integrating tabu search with classical optimization methods provides great potential for improving the results of a disruption management technique.

The last paper we have studied that uses metaheuristics is (Zhao & Guo, 2012). Here, the authors developed a new hybrid heuristic procedure based on *Greedy Random Adaptive Search Procedure (GRASP)* (Feo & Resende, 1989) and *Ant Colony Optimization(ACO)* (Dorigo, 1992; Colorni *et al.*, 1991) applied to the aircraft recovery problem. Compared with the original GRASP method, the proposed algorithm demonstrates quite high global optimization capability. The objective is to minimize the total cost of reassignments of aircraft to flights, delaying of flights and cancellations of flights. The approach also considers aircraft maintenance requirements as well as other restrictions and regulations. Computational experiments on large-scale problems show that the proposed procedure is able to generate feasible revised flight schedules of a good quality in less than 5 seconds. According to the authors, this approach was applied successfully to an airline.

3.4.1.3 Special Cases

To finish the airline recovery literature section we are going to present four papers that can be considered as describing special aircraft recovery cases. (Talluri, 1996) researches the special case of changing the aircraft type assigned for a specific flight while maintaining the feasibility of the schedule. The core of this approach is the *swap opportunity* defined by the authors as ”change of aircraft type where the new assignment is also valid”. The method is based on classifying these swap opportunities according to the number of overnight aircraft changes involved in the swap. A polynomial time algorithm is proposed by the authors. Testing was limited to a single instance.

Another special case is the problem of optimization under the *Ground Delay Program (GDP)* of the USA aviation authority (FAA). The problem can be defined as follows. Given a set of arriving flights and a set of landing slots, the landing of the incoming flights must be adjusted in order to minimize the maximum delay of outgoing flights. This, of course, might have an impact on the airline schedule and that is the reason why it is related to disruption management. In (Luo & Yu, 1997a) and (Luo & Yu, 1997b) the authors model it as an assignment problem with side constraints and developed a heuristic to solve the landing assignment problem.

The last special case paper applies *Grey Programming* (Liu & Guo, 1992) to the aircraft recovery problem. (Zhao & Zhu, 2007) transforms the model of (Jarrah *et al.*, 1993) to the concepts of grey programming. Their approach includes surplus aircraft and minimizes delays, cancellations and costs. The feasibility and effectiveness of the model and method are verified by the simulation results although, in our opinion, with a very small case of 20 flights.

3.4.2 Literature on Crew Recovery

Most of the literature on crew recovery uses traditional *OR* methods or hybrid ones, i.e., *OR* together with search heuristics. As such, it does not make sense to divide them according to the method used as we have done regarding the aircraft recovery literature, presented in the previous section. Looking at the twelve papers we have studied, we can see that near 42% of them make a strong assumption, i.e., they assume the flight schedule will not change during the crew recovery process and, as such, that the aircraft recovery has already been done. This assumption is one of the major drawbacks of the proposals presented here. The rest of them are able to deal with changes to the flight schedule, namely, flight cancellations or delays. Even though this does not overcome the drawback of the other literature, since it does not recover the aircraft part, it allows an important flexibility during the crew recovery process. Another drawback of the crew recovery literature, is that most approaches are only able to recover flight crew (pilots) problems or, if they are able to deal with cabin crew ones, they do that in a very limited way. In the next sub-sections we detail the literature according to the approaches that assume a fixed flight schedule and the ones that allow flight cancellations and delays, respectively.

3.4.2.1 Approaches Assuming a Fixed Flight Schedule

Representing the problem as a multi-commodity integer network flow model and using a depth-first branch and bound search heuristic, is the proposal presented in (Wei *et al.*, 1997). According to the authors, one prominent feature of the algorithm they proposed is that business rules are used to bound the solutions (number of modified pairings, number of impacted flights, etc.) and that, compared with traditional *OR* algorithms, is very flexible in terms of meeting some of the business requirements, such as partial solutions and multiple solutions. The goal is to return as soon as possible to the original schedule while minimizing the operational cost. To reduce the size of the problem, they include only a fraction of the schedule into the recovery problem, by defining a schedule time window. After performing extensive testing on problems with realistic sizes, they conclude that the algorithm is efficient enough for practical applications.

(Stojkovic *et al.*, 1998) also represents the problem as an integer non-linear multi-commodity network flow, which is decomposed into a set partitioning master problem and a shortest path with

constraints subproblem. To get the integer solutions, the authors use a column generation method embedded in a branch and bound search tree. The solutions found cover, at a minimal cost, all flights in a given time period with available crew while minimizing the disturbances of crew members. To generate modified pairings for selected crew members, both the classical crew pairing problem and the problem of constructing personalized monthly assignments is treated simultaneously. It is important to point out that, for some airlines (like TAP Portugal), regulations do not allow to generate completely a new personalized monthly assignment, after being released to the crew members, making this approach not suitable for some airline companies. According to the authors, solutions to one-day period test scenarios were found within half a minute. Regarding the seven-days period test scenarios, solutions took between 4 to 20 minutes to be found.

A proposal that combines exact optimization methods and metaheuristics, is present in (Guo, 2004). Here, the authors present two solution methods. One is an exact method, based on a column generation (CG) type of procedure with LP relaxation of the set partitioning problem. The other is an hybrid one combining a genetic algorithm with a local search. The main focus of the paper is on a procedure called *strategy mapping*. The strategy mapping prioritizes the alternative solutions by evaluating different criteria, such as additional cost for recovering the schedule, solution time, number of crew members that need to be notified and the number of disturbances to crew. A solution strategy is a combination of the two solution methods and relevant parameters. A case study involving data from an European airline with several home bases is presented. The authors compare the two solution method strategies and conclude that the genetic algorithm one is preferred to the column generation method, producing an acceptable solution within 3 minutes.

An approach well suited for most of the European airlines is presented by (Nissen & Haase, 2006). The authors present a new duty-period-based formulation for the crew recovery problem that is different from the earlier published modeling approaches based on pairings[5]. According to the authors, European airlines rely on a system of *fixed crew salaries*, whereas North American airlines use a system called *pay-and-credit*, where a crew member's actual duty and flight time determines his salary. In the latter case, the airlines will most likely aim to minimize costs incurred by changing the schedule and, as such, the use of pairings is essential for calculating these costs. On the former case, the aim is to adhere as closely as possible to the original schedule and, as such, pairing calculations are not essential. To deal with duty periods instead of pairings has the advantage of having shorter rescheduling horizons and, thus, smaller problems. The authors use a new type of resource constraints to efficiently cover the various labor regulations and a branch-and-price solution method. The approach was tested on several scenarios covering realistic disruptions. Results show that the solution method is capable of providing solutions within the short period of time available to a rescheduler after a disruption occurs.

The last paper that uses fixed flight schedule as an input is (Medard & Sawhney, 2007). Typically, at the planning stage, crew scheduling has two steps: create working patterns or pairings and, then, assign them to individual crew members. In this paper the authors integrate both steps in one to solve critical crew recovery problems arising on the day of operations. The optimization model proposed is formulated as a set covering model which is solved using column generation. The columns are generated by finding shortest paths using either a depth-first search strategy or a reduced cost column generator. The authors conclude that the latter performs worse and, as such, needs to be refined. Single home base tests are solved in approximately one and half minute, while the multi-base tests take several minutes to be solved.

[5] Duty periods include one or more flights, starting with the crew sign-on, and are compliant with all regulations required.

3.4.2.2 Approaches Capable of Dealing with Flight Cancellations and Delays

In this sub-section we are going to present approaches that do not require a fixed flight schedule as an input. Despite this fact, it does not mean that these approaches are able to recover the aircraft part while recovering the crew part. It means that they are able to deal with flight cancellations and delays during the crew recovery process, while keeping the aircraft itineraries.

To the best of our knowledge, (Johnson *et al.*, 1994) is the first published work regarding crew recovery. The problem is formulated as a set covering problem with decision variables allowing flight cancellations, determining the number of extra-crew and forcing crew to stay at base. In the formulation the authors only consider the recovery of flight crew pairings when a single flight is delayed at a single airport. All pairings to be considered in the crew recovery problem are generated *a priori* from a time-line network and the set covering problem is solved using MINTO (mixed integer optimizer) (Nemhauser *et al.*, 1994). Three small tests are described using data from Northwest airlines.

The problem formulation of (Johnson *et al.*, 1994) is used by (Lettovsky *et al.*, 2000). Here, the authors develop, implement and test a new solution framework that provides, in almost real time, a recovery plan for reassigning crews to restore a disrupted crew schedule. Preprocessing techniques are used to extract a subset of the schedule for rescheduling. A fast crew pairing generator constructs feasible continuations of partially flown crew trips and an efficient tree-based data structure is used for storage and data access. The crew model is solved with LP relaxation and branch and bound strategies, that consider cancellation and extra-crew variables. The authors consider the original planned schedule as optimal and, as such, their method disturbs as little as possible that plan. Computational results using a schedule from a major USA airline were performed. The results show that medium-sized disruptions to the crew schedule can be handled within an acceptable running time. The authors also conclude that further research is required to handle large-scale disruptions.

A report of a successful implementation of a crew recovery decision support system called *CrewSolver* for Continental Airlines is presented in (Yu *et al.*, 2003). According to the authors, *CrewSolver* generates globally optimal, or near optimal, crew recovery solutions and that, since its implementation, was able to deal successfully with several high-profile events, including the September 11th terrorist attacks. The *CrewSolver* uses the depth-first search procedure developed by (Wei *et al.*, 1997). The authors claim that the generated solutions support the disrupted flight schedule at the lowest cost possible while maintaining a high quality of life for its pilots and flight attendants. They also claim that several other airlines have acquired the *CrewSolver* decision support system.

The work presented in (Stojkovic & Soumis, 2001) is an extension of the authors previous work (Stojkovic *et al.*, 1998), detailed by us on the previous sub-section. The main extension is the possibility of delaying the flights without changing the aircraft schedule itineraries. The problem is mathematically formulated as an integer non-linear multi-commodity network flow model with time windows and additional constraints. To solve the problem, a Dantzig-Wolfe decomposition combined with a branch and bound method was used. The master problem comprises the flight covering constraints and a new set of flight precedence constraints. Sub-problems consisting of time-constrained shortest-path problems with linear time costs are solved by a specialized dynamic programming algorithm. The objectives are to minimize the number of canceled flights, the total delay of all flights, and the number of pilots whose planned activities for the next day of operations must be changed as a result. The solutions may include the use of reserve pilots. The approach

provides limited support or imposes several restrictions, when used to solve cabin crew disruptions, and that is a drawback. The approach was tested on three problems. The largest one has 59 pilots and 190 flights. All problems are tested with and without reserve pilots, allowing delaying flights, and with a fixed flight schedule. The results are encouraging both in terms of computing times and solution quality.

The previous work was further extended in (Stojkovic & Soumis, 2005) by including the capability of dealing with multiple crew members. This is achieved by using a number of copies of each flight corresponding to the number of crew members required. A set of constraints ensure that the departure times for all copies of each flight are added to the model. The solution process is very similar to the previous work described in (Stojkovic & Soumis, 2001). The authors tested three different models: one corresponding to that from the previous work with strict flight covering constraints, another in which there is a linear cost for missing crew members and the last one, with a cost for each flight with a missing crew. They demonstrate that in the second and third model, there are substantial improvements. However, for large problems, the computational solutions take more than one hour to be found and that is a major drawback for these models to be used in real operational scenarios.

A decision support tool for flight crew recovery for airlines that adopt the *hub-and-spoke* network is presented in (Abdelghany *et al.*, 2004). According to the authors, the major advantage of this tool is that "proactively recovers crew problems ahead of time before their occurrence" giving a wide flexibility to react to different operation scenarios. It solves for the most efficient crew recovery plan with the least deviation from the originally planned schedule. The tool adopts a rolling approach in which a sequence of optimization assignment problems is solved such that it recovers flights in chronological order of their departure times. In each assignment problem, the objective is to recover as many flights as possible while minimizing total system cost resulting from resource reassignments and flight delays. As a test case, the authors use one disruption scenario from the operations of a major USA airline that includes 18 disrupted crew members and 121 candidate crew members. The number of affected flights is not provided by the authors. The recovery problem was solved in 2 minutes.

Finally, the last work regarding crew recovery is presented in (Zhao *et al.*, 2007). The authors transform the problem formulation of (Abdelghany *et al.*, 2004) into a grey programming (Liu & Guo, 1992) model. Linear decision variables for departure and arrival times of flights and the linear parameter defining the ready time for crew in (Abdelghany *et al.*, 2004) model, are defined as grey variables. A local search heuristic is used to solve the problem. The basic framework of the implementation is the same as (Zhao & Zhu, 2007) that deals with aircraft recovery. The authors use the same small test case used in the aircraft recovery paper, without giving any further details.

3.4.3 Literature on Partial-Integrated Recovery

As we stated before, the number of proposals that are able to recover at least two of the parts of the disruption problem are increasing substantially, specially in the 2008 to 2013 period. We classify these proposals as *partial-integrated recovery (PIR)*. In this sub-section we provide details about the ten research work we have studied, grouping them by the proposals that are able to recover aircraft and crew and aircraft and passenger, respectively. Not surprisingly, most of the proposals

are for aircraft and passenger recovery, since it is less complex to integrate both problems than the aircraft and crew ones.

3.4.3.1 Aircraft and Crew Recovery

In (Gao, 2007) the authors study and propose an approach to integrate fleet assignment with crew scheduling during the airline planning process, so that the solutions are more robust to real time operations. Additionally, the authors apply this approach to recovery during daily operation, in a process they call *integrated recovery*. According to them, the major challenge is to stop the ripple effect caused by disruptions and, for that, the main strategy they used was to define different recovery sets for schedule change, aircraft rerouting, and crew rerouting. They propose a new integrated recovery model and Benders decomposition solution. In the integrated recovery model, the duty flow model is combined with fleet assignment in the master problem. The duty network is then built in a dated version and is crew specific because of the features of recovery. Instead of enumerating aircraft routings, a multi-commodity network flow model is adopted to model the aircraft maintenance routing, which can reduce the number of variables without losing maintenance considerations. In the Benders decomposition method, feasibility cuts coming from aircraft maintenance routing problems are generated and returned to the master problem. The authors do not present computational tests regarding the integrated recovery approach. However, regarding the tests they have performed for the same approach applied during the planning phase, the authors conclude that the approach is very efficient in solving industrial size problems.

To the best of our knowledge, we were the first to propose a multi-agent system (MAS) approach to disruption management. In our previous work (Castro & Oliveira, 2007) we proposed a MAS architecture that included agents (called *managers*) responsible for each part of the problem, i.e., aircraft, crew and passengers. Each manager had a team of *specialist* agents that implement several heterogeneous problem solving algorithms based on metaheuristics. For example, we have implemented *simulated annealing* (Kirkpatrick *et al.*, 1983) and *hill climbing* to solve the aircraft and crew recovery problem. Each manager requests several candidate solutions to their specialists and, then, chooses the best integrated solution according to its preferences. This approach was tested with real data from TAP Portugal and compared with the solutions found by the human operators on the Airline Operations Control Center (AOC).

(Abdelghany *et al.*, 2008) propose a proactive recovery tool called *DSTAR* that has the goal of integrating aircraft and crew (flight and cabin) recovery. A rolling horizon modeling framework, which integrates a schedule simulation model and a resource assignment optimization model, is adopted for this tool. The schedule simulation model projects the list of disrupted flights in the system as function of the severity of anticipated disruptions. The optimization model (mixed integer programming) examines possible resource swapping and flight re-routing to generate an efficient schedule recovery plan that minimizes flight delays and cancellations. The authors describe an application scenario where *DSTAR* saves 8.7% of the total delay. The approach of the paper is very promising when considering larger disruptions, which are foreseeable a number of hours ahead.

3.4.3.2 Aircraft and Passenger Recovery

(Bratu & Barnhart, 2006) presents airline schedule recovery models and algorithms that simultaneously develop recovery plans for aircraft and passengers by determining which flight leg departures

to postpone and which to cancel. The objective is to minimize airline operating costs and estimated passenger delay and disruption costs. The approach considers crew regulations for reserve crew but the recovery of the disrupted crew is not considered. The generated recovery plans suggest flight departure times and flight cancellations, assigning aircraft and reserve crews, if necessary, to flight legs, while complying with crew regulations and satisfying aircraft maintenance requirements. They propose two optimization models each minimizing airline operating costs jointly with some measure of passenger costs. The passenger costs considered are passenger disruption costs in the *disrupted passenger metric model (DPM)* and passenger delay costs in the *passenger delay model (PDM)*. In the latter the delay costs are modeled exactly by explicitly modeling disruptions, recovery options and delay costs, whereas in the former, delays costs are only approximate. The authors developed an AOCC simulator to evaluate the recovery models. They found that *PDM* cannot be solved in real time for day-long decision windows and days with relatively high levels of disruption, whereas *DPM* is fast enough to be used by operations controllers to recover from airline irregularities in real time. *DPM* could nonetheless be used to solve smaller instances that can be partitioned. For 3 days of operations with different levels of disruption, using *DPM* they were able to generate solutions with noticeable reductions in passenger delays and disruptions. Considering the results, the authors conclude that better operation decisions can be generated in real time to reduce passenger delays and disruptions, while recovering airline resource schedules and controlling airline operating costs.

A passenger recovery alternative involving ground transportation, is presented by (Zhang & Hansen, 2008). The authors proposed an integer with non-linear objective function model to determine flight cancellations and inter-modal substitutions in the case when a hub airport encounters a relatively severe capacity shortfall. The model takes into account the net delay saving of canceling one particular flight over other flights in the schedule. The objective is to minimize passenger and operational costs. Results show that *real-time inter-modal substitution (RTIMS)* can save about 8% to 14% of the disruption cost compared to the model without *RTIMS*. According to the authors, there are several issues that need further consideration. First, variability and uncertainty of the ground transportation times is a potentially important issue. Second, a more complete model of passenger re-accommodation on subsequent flights is required. Third, the potential to combine *RTIMS* with flight re-ordering should be considered. After performing tests, the authors conclude that the use of *RTIMS* is a promising topic for future research to alleviate airlines disruption cost and reduce passenger delay.

During the ROADEF[6] 2009 Challenge for *Disruption Management for Commercial Aviation* sponsored by AMADEUS[7], (Rodrigo Acuna-Agost & Gueye, 2009) proposed a model for solving the flight, aircraft and passenger recovery problem. They proposed a mixed integer programming formulation and a solution method that uses a statistical analysis called *Statistical Analysis of Propagation of Incidents (SAPI)*, for concentrating the search on probable good solutions. According to the authors, their method obtained very good results in the competition, showing to be quite effective and viable to be applied in real problems.

According to (Jafari & Zegordi, 2010), the problem of recovering disrupted passengers after aircraft recovery decisions, has not been explicitly considered in most previous aircraft recovery models. As such, they present an assignment model for airline schedule recovery which recovers both aircraft and disrupted passengers, using a rolling horizon time framework. Their model

[6] La société française de Recherche Opérationnelle et d'Aide à la Décision

[7] An IT Company with solutions for airlines (http://www.amadeus.com/airlineit/index.html)

examines possible flight retiming, aircraft swapping, overflying, ferrying, utilization of reserve aircraft, cancellation and passenger reassignment to generate an efficient schedule recovery plan. The model ensures that the schedule returns to normal within a certain time and the objective is to minimize operational recovery aircraft cost, cancellation and delay cost as well as disrupted passenger cost. They use mixed integer and *LINGO 8.0*[8] to solve the problem. It is important to point out that this model does not generate itineraries for the disrupted passengers. The model was tested using a data-set with two disruption scenarios. The computational results show that it is capable of handling the integrated aircraft and passenger recovery problem successfully. Later, the same authors proposed a solution to the same problem in (Zegordi & Jafari, 2010), that uses the Ant Colony (ACO)(Colorni *et al.*, 1991; Dorigo, 1992) algorithm.

In (Eggenberg *et al.*, 2010) the authors present a modeling framework that allows the consideration of operational constraints within a column generation (CG) scheme. They introduce the general concept of recovery network, generated for each individual unit of the problem, and show how unit-specific constraints are modeled using resources. The concept is illustrate by solving the aircraft recovery problem with maintenance planning, with some insights into applying the model to the passenger recovery problem. According to the authors, it is possible to apply the same model to the crew recovery problem although nothing is shown in the paper. However, even in the case of applying the model to the three dimensions, we still do not have an integrated and simultaneous resolution of the problem. The output of a sub-problem is the input of another. Similar to what happens with Petersen's method (Petersen *et al.*, 2010).

The last paper classified by us in the *partial-integrated* category is (Bisaillon *et al.*, 2010). The authors introduce a large neighborhood search heuristic for an airline recovery problem combining fleet assignment, aircraft routing and passenger assignment. Given an initial schedule, a list of disruptions, and a recovery period, the problem consists in constructing aircraft routes and passenger itineraries for the recovery period that allow the resumption of regular operations and minimize operating costs and impacts on passengers. The heuristic alternates between construction, repair and improvement phases, which iteratively destroy and repair parts of the solution. The aim of the first two phases is to produce an initial solution that satisfies a set of operational and functional constraints. The third phase then attempts to identify an improved solution by considering large schedule changes while retaining feasibility. The whole process is iterated by including some randomness in the construction phase so as to diversify the search. This work won the first prize of the ROADEF 2009 Challenge.

3.4.4 Literature on Integrated Recovery

We define an integrated recovery process as that which is able to recover all problem dimensions (aircraft, crew and passenger) separately but not simultaneously. Typically, they are solved sequentially and in the following order: aircraft, crew and passenger (see Definition 3.11). Naturally, this solving order imposes an importance factor to the dimensions.

Unfortunately, we did not find many works that use this kind of approach. In the 1996 to 2013 period we found only three approaches with only one new work every six years. A possible explanation for the lack of research proposals using an integrated approach, is the complexity of the problem when trying to obtain an integrated solution that includes the three dimensions, specially

[8] LINGO is a software tool designed to efficiently build and solve linear, nonlinear, and integer optimization models

when using a monolithic method or process. We believe that a non-monolithic method, e.g., a distributed one supported by an adequate decision mechanism, allows to better solve this problem by diminishing its complexity and without imposing importance factors to the dimensions (see our hypotheses in Section 1.3.2).

The first proposal of a truly integrated approach, although only parts of it are implemented, was presented in the Ph.D. thesis of (Lettovsky, 1997). Most of the work of this dissertation focuses on the crew recovery problem, which is the only one that was implemented. However, the author first formulated the *Airline Integrated Recovery (AIR)* problem which has three parts: crew assignment, aircraft routing, and passenger flow. A new decomposition scheme is proposed that includes as its master problem the *Schedule Recovery Model (SRM)*. This model provides a cancellation and delay plan that satisfies imposed landing restrictions and assigns equipment type. Once this master problem is solved, the three problems for crew, aircraft, and passengers decouple. The operational plan for each equipment type is formulated in two separate subproblems, an *Aircraft Recovery Model (ARM)* and a *Crew Recovery Model (CRM)*. Either an aircraft and a crew are found for each flight leg or the flight is canceled. A *Passenger Flow Model (PFM)* subproblem then finds new itineraries for disrupted passengers. The solution algorithm is derived applying Benders' decomposition algorithm to a mixed-integer linear programming formulation for the problem. This hierarchical and sequential structure of the framework resembles the current manual disruption management process used by the majority of the airlines.

(Kohl *et al.*, 2004) reports on the experiences performed during the research and development of project DESCARTES (a large scale project supported by EU) on airline disruption management. The current (almost manual) mode of dealing with recovery is presented. They also present the results of the first prototype of a multiple resource decision support system. The tool includes dedicated solvers for each part of the problem, i.e., a *Dedicated Aircraft Recovery Solver (DAR)*, *Dedicated Crew Recovery Solver (DCR)* and *Dedicated Passenger Recovery Solver (DPR)*. Additionally, an *Integrated Recovery Layer (IRL)* was included. The goal of *IRL* was to provide an integrated solution that included the three dimensions. For that, the authors propose two approaches. One, called *Integrated Sequential Recovery (ISR)* that uses the dedicated solvers *DAR*, *DCR* and *DPR* as *black boxes* for solution generation and evaluation. The other is called *Tailored Integrated Recovery (TIRS)* that has a mathematical model incorporating all dimensions instead of having a model for each dimension. According to the authors, the integrated recovery was an active research issue but no results were obtained.

The last work we have studied is (Petersen *et al.*, 2010). The authors consider an integrated approach that is able to recover the flight schedule, aircraft rotations, crew schedule, and passenger itineraries in a tractable manner. According to the authors, they were the first to present computational results on the fully integrated airline recovery problem. Those results show that the integrated approach can substantially improve the solution quality over the incumbent sequential approach. An optimization approach is used to represent and solve the problem that resembles the one used by (Lettovsky, 1997). A Benders' decomposition scheme is used to decompose the problem. The *Schedule Recovery Model (SRM)* is the master problem with linking variables to be passed in to the subsequent subproblems, i.e., *Aircraft Recovery Model (ARM)*, *Crew Recovery Model (CRM)* and *Passenger Recovery Model (PRM)*. An important limitation is that the *CRM* only considers flight crew and is not able to recover disrupted cabin crew members. The authors consider their approach fully integrated because they are considering the three (or four) dimensions of the problem: aircraft (flight), crew and passengers. The resolution algorithm used applies a kind

of backtracking. Nevertheless, a sub-problem resolution order is naturally imposed by the algorithm making some sub-problems more important than others. In this case, the aircraft problem is more important than the crew problem and both are more important than the passenger problem, that is, the output of one is the input of another. This approach was tested using data from a USA airline with a dense flight network. It is shown that in several instances an integrated solution is delivered in a reasonable runtime.

3.4.5 Literature on Simultaneously-Integrated Recovery

From the study of the existing approaches classified as *partial-integrated* or *integrated*, it is possible to see that there is an hierarchy regarding the dimensions of the problem and that a sequential problem solving process is used. Even in the work of (Petersen *et al.*, 2010), although not as much as in the others due to the *backtracking* approach used in the algorithm, that hierarchy and sequence is present. In our opinion, this makes some dimensions more important than others.

As of the time of writing this thesis we did not find any proposed work that is able to recover all three dimensions simultaneously, without imposing an hierarchy and/or importance to any of the dimensions, i.e., a *Simultaneously-Integrated Recovery* approach.

To the best of our knowledge, the approach we proposed in this thesis is the first one that takes care of each sub-problem or dimension in parallel and simultaneously, making them equally important. Additionally, we also believe that our approach is the first one that can be classified as an *Automatic or Semi-Automatic System (ASAS)* since it is able to represent all the functions and roles of the AOCC, working automatically and autonomously, requiring a minimum of human intervention to work. Chapters 6 and 7 of this thesis sustain this claim.

Table 3.1: Comparative summary regarding operations recovery

Author(s)	**Main Strategies/Objectives**	**Main Model/Solver**	**Rec.**	**Cat.**
(Castro & Oliveira, 2011; Castro *et al.*, 2012)	Aircraft, Crew and Passenger Recovery. Minimize delays and costs. Considers several requirements and restrictions.	Multi-Agent System. Negotiation as Decision Mechanism. Learning. Simulated Annealing, Hill Climbing, Shortest-path.	SIR	ASAS
(Zhao & Guo, 2012)	Minimize total cost of assignment, cancellation and delays. Considers maintenance requirements and other regulations.	Greedy Random Adaptive Search Procedure (GRASP) and Ant Colony Optimization (ACO).	AR	DSS
(Wu & Le, 2012)	Aircraft Recovery. Minimize Total Cost of Assignment, Cancellation and Delays, Considers maintenance and other regulations.	Time-space Network; Iterative Tree Growing with Node Combination Model.	AR	MALG
(Bisaillon *et al.*, 2010)	Aircraft and Pax Recovery. Minimize Operational Costs and Impact on Passengers	Large Neighborhood Search (LNS) Heuristic.	PIR	MALG
(Eggenberg *et al.*, 2010)	Aircraft Recovery and insights about application to Pax Recovery. Crew recovery supported but not modeled. Operational and delay costs.	Column Generation (CG) scheme.	PIR	MALG
(Jafari & Zegordi, 2010)	Airline Schedule Recovery, Aircraft and Passenger Recovery. Minimize Operational Aircraft Recovery, Cancellation, Delay and Disrupted pax costs.	Mixed Integer; LINGO 8.0.	PIR	MALG
(Zegordi & Jafari, 2010)	Airline Schedule Recovery, Aircraft and Passenger Recovery. Minimize Operational Aircraft Recovery, Cancellation, Delay and Disrupted pax costs.	Ant Colony (ACO) algorithm	PIR	MALG
(Petersen *et al.*, 2010)	Aircraft, Crew and Pax recovery. Aircraft, Crew and Pax costs considerations including pax goodwill costs.	Multi-commodity Network Flow Model. Mixed-Integer Programming. Benders Decomposition. Column Generation (CG). LP Relaxation.	IR	MALG

Table 3.1: Comparative summary regarding operations recovery

Author(s)	**Main Strategies/Objectives**	**Main Model/Solver**	**Rec.**	**Cat.**
(Rodrigo Acuna-Agost & Gueye, 2009)	Flight, Aircraft and Pax Recovery. Minimize Operational Costs and Impact on Passengers.	Mixed Integer Programming plus Statistical Analysis of Propagation of Incidents (SAPI).	PIR	MALG
(Abdelghany *et al.*, 2008)	Resource reschedule; Resource swapping; Aircraft, pilots and cabin crew; Minimize cost resource assignments, cancellations and delays.	Mixed Integer Programming	PIR	DSS
(Zhang & Hansen, 2008)	Pax Ground transportation. Delays and Cancellations. Minimize Passenger and Operational Costs.	Integer with non-linear objective function.	PIR	MALG
(Liu *et al.*, 2008)	Flight connections and swaps; Total flight delay; Cancellations; Assignment. Multi-fleet.	Multi-objective genetic algorithm (Metaheuristics)	AR	MALG
(Gao, 2007)	Recovery sets for schedule change, aircraft rerouting and crew rerouting. Aircraft maintenance considerations. Aircraft/flight, crew, delay and cancellation costs.	Benders Decomposition; Multi-commodity Network Flow Model.	PIR	MALG
(Yang, 2007)	Minimize flight schedule modifications, delay, cancellation and swap costs.	Tabu search and time-space network flow model with side constraints.	AR	—
(Zhao & Zhu, 2007)	Surplus aircraft; Delay; Cancellations; Cost.	Grey programming; Local search heuristic.	AR	MALG
(Eggenberg *et al.*, 2007)	Recovery plans; Cancellations; Flight, delay, maintenance cost. Aircraft maintenance considerations.	Set partitioning; Column generation; Resource constraint shortest path.	AR	MALG
(Zhao *et al.*, 2007)	Flight schedule modifications; Crew, Flight delay cost; Individual roster	Grey programming; Local search heuristic.	CR	MALG
(Castro & Oliveira, 2007)	Crew and aircraft swap, reserve crew and aircraft; Crew cost; Individual roster.	Multi-agent system; Hill Climbing and Simulated annealing.	PIR	ASAS

Table 3.1: Comparative summary regarding operations recovery

Author(s)	**Main Strategies/Objectives**	**Main Model/Solver**	**Rec.**	**Cat.**
(Medard & Sawhney, 2007)	Assumes recovery flight schedule first; Illegal crew, uncovered flights and affect crew; Individual roster.	Set covering model and column generation plus Depth-first search or reduced cost column generator.	CR	MALG
(Liu *et al.*, 2006)	Swap or delay flights. Minimize delay costs, swap costs and aircraft assignment costs.	Multi-objective genetic algorithm (Metaheuristics)	AR	MALG
(Bratu & Barnhart, 2006)	Delay, cancel, assign reserve crew and aircraft. minimize operating costs, passenger delay and disruption costs.	Flight schedule network	PIR	MALG
(Andersson, 2006)	Cancellations, delays and aircraft swap.	Tabu and Simulated Annealing (Metaheuristics)	AR	MALG
(Nissen & Haase, 2006)	Assumes recovery flight schedule first; Duty-based formulation; Modifications original schedule; Individual roster	Branch-and-price; Set covering; Resource constrained shortest path.	CR	MALG
(Stojkovic & Soumis, 2005)	Departure delays; Reserve pilots; Modifications, uncovered flights, flight delays; Individual roster. Multiple crew members.	Multi-commodity network flow; Column generation.	CR	MALG
(Løve *et al.*, 2005)	Cancellations, delays, swaps, ferrying. Revenue minus costs	Metaheuristics	AR	MALG
(Andersson & Varbrand, 2004)	Cancellations, swap and aircraft swap	Set packing problem with generalized upper bound (GUB) constraints; Lagrangian relaxation-based heuristic and Dantzig-Wolfe decomposition.	AR	DSS
(Abdelghany *et al.*, 2004)	Deadheading, stand-by, swap, flight delay costs; Individual roster. Hub-and-spoke airline network.	Mixed-integer program;	CR	DSS

Table 3.1: Comparative summary regarding operations recovery

Author(s)	**Main Strategies/Objectives**	**Main Model/Solver**	**Rec.**	**Cat.**
(Guo, 2004)	Assumes recovery flight schedule first; Stand-by, modifications, operating costs; Individual roster	Set partitioning problem; Column generation with LP relaxation or Hybrid heuristic based in a genetic algorithm with a local search.	CR	MALG
(Kohl *et al.*, 2004)	Flight swaps, cancellations, crew swaps, stand-by, up/downgrading crew; Passenger delay costs at destination, value of passenger based on the booked fare class and frequent flyer information.	Dedicated aircraft solver (Extension Local Search Heuristic); Dedicated crew solver (Differential column-generation/constraint integer problem); Dedicated passenger solver (multi-commodity flow problem); Integrated recovery layer (Intelligent messaging system).	IR	DSS
(Yu *et al.*, 2003)	Cancellations; Deadheading, modifications, uncovered flight costs	Depth-first search; *CrewSolver* optimization.	CR	DSS
(Rosenberger *et al.*, 2003)	Delay and cancellation	Set partitioning model; Preprocessing heuristic; CPLEX 6.0.	AR	MALG
(Andersson, 2001)	Flight cancellation and delays. Multi-fleet aircraft swaps. Maximize revenue.	Mixed Integer Multi-commodity network flow with side constraints; Lagrangian relaxation, Dantzig-Wolfe decomposition, Column Generation, Tabu Search.	AR	DSS
(Bard *et al.*, 2001)	Minimize delay and cancellation costs.	Integer minimum cost flow model with additional constraints.	AR	MALG
(Thengvall *et al.*, 2001), (Thengvall *et al.*, 2003)	Cancellations; Multi-fleet; Revenue minus cost, Hub closure considerations.	Three mixed-integer program models.	AR	MALG
(Stojkovic & Soumis, 2001)	Minimize number canceled flights, total delay of flights and number pilot modifications. Individual roster. Single crew member.	Multi-commodity network flow with additional constraints; Column generation.	CR	MALG

Table 3.1: Comparative summary regarding operations recovery

Author(s)	**Main Strategies/Objectives**	**Main Model/Solver**	**Rec.**	**Cat.**
(Lettovsky *et al.*, 2000)	Cancellation; Pairing, cancel flight costs.	Set covering with decision variables; LP Relaxation and Branch-and-Bound	CR	MALG
(Thengvalla *et al.*, 2000)	Cancellations, swaps, delays; Revenue minus costs	Integer programming; LP relaxation with heuristic	AR	MALG
(Stojkovic *et al.*, 1998)	Assumes recovery flight schedule first; Pairing, Deadheading, undercover costs; Individual roster	Set partitioning problem. Integer non-linear multi-commodity flow network problem; Columns generation, branch-and-bound.	CR	MALG
(Luo & Yu, 1997b)	Delayed flights	Assignment problem with side constraints; Heuristic	AR	MALG
(Lettovsky, 1997)	Cancellation, delays, equipment assignment; Maximizes total profit.	Linear mixed-integer programming. Benders decomposition. Set covering. Branching strategies.	IR	MALG
(Wei *et al.*, 1997)	Assumes recovery flight schedule first; Pairing cost. Minimizes operational cost.	Set covering problem. Integer multi-commodity network flow problem; Depth-first branch and bound search.	CR	MALG
(Arguello *et al.*, 1997)	Cancellations and delays; Flight route augmentation, partial route exchange; Route, cancellation and reassign costs.	Metaheuristics (GRASP – Greedy Randomized Adaptive Search Procedure)	AR	MALG
(Luo & Yu, 1997a)	Number delayed flights under GDP (Ground Delay Program)	Assignment problem with side constraints; Heuristic	AR	MALG
(Cao & Kanafani, 1997a,b)	Cancellations and delays; Aircraft swap of different types. Maximize revenue minus costs	Minimum cost network flow; Network flow algorithms.	AR	MALG
(Yan & ping Tu, 1997)	Cancellations and delays; Ferrying, Multi-fleet; Costs minus revenues	Network flow model with side constraints; Lagrangian relaxation with subgradient method, Lagrangian heuristic.	AR	MALG

Table 3.1: Comparative summary regarding operations recovery

Author(s)	**Main Strategies/Objectives**	**Main Model/Solver**	**Rec.**	**Cat.**
(Clarke, 1997, 1998)	Cancellations; Multi-fleet; Costs minus revenues. Maintenance, crew availability and slot allocation considerations.	Set partitioning, Column generation, extra constraints; Tree-search heuristic and a set packing-based optimal solution.	AR	MALG
(Yan & Dah-Hwei, 1996)	Cancellations and delays; Ferrying. Costs minus revenues	Minimum cost network flow; Network flow algorithms.	AR	MALG
(Talluri, 1996)	Multi-fleet; Swaps when exchanging aircraft type.	Classifies swap opportunities; Polynomial time algorithm.	AR	MALG
(Mathaisel, 1996)	Cancellations; Revenue loss, operating cost	Minimum cost network flow; Network flow algorithms.	AR	DSS
(Rakshit *et al.*, 1996)	Cancellations; Delay, swap and ferrying.	Minimum cost network flow; Network flow algorithms.	AR	DSS
(Teodorovic & Stojkovic, 1995)	Cancellation; Crew considerations; Minimize total passenger delays.	Heuristic; FIFO; Dynamic Programming.	AR	DSS
(Johnson *et al.*, 1994)	Pairing, stand-by, deadheading costs; Cancellations.	Set covering problem with decision variables; *MINTO* (Nemhauser *et al.*, 1994) (mixed integer optimizer)	CR	MALG
(Jarrah *et al.*, 1993)	Cancellations and total passenger delays. Swap, ferrying and re-timing.	Minimum cost network flow; Network flow algorithms.	AR	DSS
(Teodorovic & Stojkovic, 1990)	Cancellation and total passenger delays	Heuristic	AR	MALG
(Teodorovic & Guberinic, 1984)	Total passenger delays; Reassign and re-timing the flights	Heuristic; Branch and Bound.	AR	MALG

3.5 Automated Negotiation

One of the main contributions in this thesis is a negotiation protocol that is generic enough to be used in different types of environments. In Chapter 6 we describe in detail the protocol, called *Generic Q-Negotiation (GQN)* and, later, we use it as a decision mechanism included in the multi-agent system we have developed for disruption management in airline operations control (see Chapter 7).

In the literature it is possible to find several proposals regarding protocols for automated negotiation. It is not our intention in this section to do an exhaustive search and comparison between all the existing works. Since we use automated negotiation in our study, our intention is to give the reader an idea and brief comparison of what researchers are doing in this domain.

In table 3.2 we refer a few protocols that have some characteristics that are similar to the protocol we presented in Chapter 6. We left out protocols that use *game-theory* and *argumentation* as their theoretical approach. The interested reader can find good papers on these subjects. For example, (Jennings *et al.*, 2001) states very clearly the difference between *game-theory*, *heuristics* and *argumentation-based* approaches to negotiation and (Rahwan *et al.*, 2004) presents one of the first survey on the subject. More recently, (Dimopoulos & Moraitis, 2010) updated this information. Regarding game-theory, (Brams, 2003) provides a good survey on the subject.

We also left out proposals related to *generic automated negotiators* or *frameworks* that include tools allowing to design several types of negotiation models including protocols and agent's strategies, since it is more difficult to compare due to the variety of protocols and agents that can be designed. For example, (Bartolini & Preist, 2001) proposed a framework that supports several models of negotiation for market-based mechanisms. Although it supports the design of several protocols, rules and strategies, when compared with our proposal, the main difference is that ours can be applied to other scenarios in addition to the market-based ones. An example of a generic automated negotiator is GENIUS (Lin *et al.*, 2011). According to the authors GENIUS "*enables alleviation of the difficulties in the design process of general automated negotiators (...) encompasses many benefits and advantages over agents that are designed for a specific domain.*" The authors conducted experiments that sustain this claim. This system also allows to negotiate against human counterparts.

Before starting to describe the related work presented in table 3.2 we want to remind that the definitions for the concepts we used to characterize the protocols were given in Section 3.2.

As stated before, the diversity of negotiation protocols available is huge. From the ones we found and presented in Table 3.2, six of them (Zeng & Sycara, 1998),(Rocha & Oliveira, 1999),(Faratin *et al.*, 2002),(Dang & Huhns, 2005),(Goradia, 2007),(Vokřínek *et al.*, 2007) are for competitive environments (Definition3.22). From those, two support agents with adaptive (Definition 3.19) characteristics. (Zeng & Sycara, 1998) proposes a negotiation protocol with a sequential decision making model called Bazaar. It is a bilateral (Definition 3.24), one-to-one (Definition 3.20), multidimensional with independent dimensions (Definition 3.21), alternating offers protocol with several rounds, with adaptive agents due to the use of Bayesian Learning, supporting only agents with full knowledge (Definition 3.23).

(Rocha & Oliveira, 1999) proposes a negotiation mechanism (called Q-Negotiation) in the context of agent-based Virtual Organization (VO) formation process. In this scenario, each organization has the objective to maximize its own profit and, for that, the negotiation process takes into account the rationality and self-interestedness of the agents. The Q-Negotiation includes a bi-

Table 3.2 Comparison of GQN with Related Work

Year and Authors	Part.	Agents	Dim.	Knowl.	Env.	Adap.
1993, (Sandholm, 1993)	Bil.	1-M	S	Full	Mixed	No
1998, (Zeng & Sycara, 1998)	Bil.	1-1	M	Full	Comp.	Yes
1999, (Rocha & Oliveira, 1999)	Bil.	1-M	M	Full	Comp.	Yes
2002, (Faratin *et al.*, 2002)	Bil.	1-1	M	Full	Comp.	No
2003, (Luo *et al.*, 2003)	Bil.	1-1	M-Interd.	Full	S-comp.	No
2004, (Aknine *et al.*, 2004)	Bil.	1-M	S	Full	Mixed	No
2005, (Dang & Huhns, 2005)	Bil.	M-M	M	Full	Comp.	No
2005, (Lopes *et al.*, 2005, 2004, 2002)	Multil.	1-M	M	Full	Mixed	No
2007, (Goradia, 2007)	Multil.	1-M	S	Full/Partial	Comp.	No
2007, (Vokřínek *et al.*, 2007)	Bil.	1-M	M	Full	Comp.	No
2008, (Crawford & Veloso, 2008)	Bil.	1-M	S	Full	S-coop.	Yes
2009, (Jain & Deshmukh, 2009)	Bil.	1-M	M-Interd.	Full	Mixed	Yes
2010, (Katsuhide Fujita & Klein, 2010)	Multil.	M-M	M-Interd.	Full	Mixed	No
2012, Our Proposal (GQN)	Bil.	1-M	M-Interd.	Full/Partial	Mixed	Yes

lateral, one-to-many, multidimensional (independent dimensions) negotiation with several rounds and qualitative feedback. The agents are able to learn (adapt) their strategies during proposal formulation, due to the inclusion of a Q-Learning algorithm (Watkins & Dayan, 1992) and they must possess full knowledge. The preferences of the agents are private. Our proposed GQN protocol is an adaptation and improvement of Q-Negotiation. When compared with our work, the main difference is that our proposal supports more environments as well as agents with full or partial knowledge and interdependent dimensions.

Unlike our proposal, the remaining four works for competitive environments do not have adaptive characteristics. (Faratin *et al.*, 2002) presents a strategy called the trade-off strategy where multiple negotiation decision variables are traded-off against one another. It operates by using the notion of fuzzy similarity to approximate the preference structure of the other negotiator and then uses a hill-climbing technique to explore the space of possible trade-offs for the one that is most likely to be acceptable. It uses an alternating sequential protocol of offers and counter-offers but only between two agents (one-to-one). The agents need to have full knowledge to propose although they have limited information about the opponent. The agent preferences are private.

(Dang & Huhns, 2005) proposes an extension to the alternating offers protocol (Osborne & Rubinstein, 1994) that allows many-to-many negotiations concurrently. According to the authors, the protocol deals effectively with issues such as negotiation consistency and decommitment risk, which arise in many-to-many negotiation. Besides this characteristic the protocol is multidimensional with independent dimension and supports agents with full knowledge. Like in the previous work, the preferences are private.

(Goradia, 2007) presents a negotiation model that is suitable for systems with distributed information, capacities and resources. Each agent represents individual components of the system and preserve the self-interest of the components they represent making decisions that maximize the expected utilities of their owners. Due to this characteristic the model supports agents with full or partial knowledge, being similar to our proposal regarding this aspect. The protocol only supports single dimension, one-to-many negotiations. According to the authors, the simulation results show that their algorithm has many desirable properties, such as, distribution, efficiency, stability, scalability, and simplicity.

A Competitive Contract Net Protocol is proposed by (Vokřínek *et al.*, 2007), that was designed to facilitate a flexible cooperation in competitive multi-agent environments. The protocol covers not only the phase of contracting the commitments, but also allows for a decommitment negotiation and contract termination. It is a multidimensional with independent dimensions, one-to-many protocol and requires the agents to have full knowledge to present proposals.

In Table 3.2 we have included one example of a protocol for semi-competitive environments and another example of a protocol for semi-cooperative environments. Luo et al (Luo *et al.*, 2003) present a bilateral, one-to-one, alternating offers protocol for semi-competitive environments where the agents need to have full knowledge to present proposals. It is multidimensional with interdependent dimensions and the authors use a fuzzy constraint based model to express buyer restrictions over simple attributes and over combinations of various attributes. The model uses fixed strategies for the buyer and seller and, as such, does not support any adaptive characteristics. However, the authors want to improve it by including alternative strategies.

(Crawford & Veloso, 2008) introduces the *Semi-Cooperative Extended Incremental Multi-Agent Agreement Problem with Preferences* (SC-EIMAPP). In the SC-EIMAPP problem, the attributes arise over time. For each attribute a set of distributed agents gain utility for agreeing in a value to assign to it (the set of possible values to be assigned to the attributes need to be known *a priori*). The semi-cooperative utility is defined as the private preferences of the agents discounted as the negotiation time increases. According to the authors, the SC-EIMAPP reflects real world problems such as scheduling and task allocation. This is a bilateral, one-to-many, single dimension protocol with adaptive agents with full knowledge to present proposals.

In our proposed *GQN* protocol there are four characteristics that, in our opinion, allow it to be applied in more application domains (making it more generic):

1. support of agents with heterogeneous behavior (covers more types of environments).
2. support of agents with full and/or partial knowledge (useful for distributed environments).
3. includes agents with adaptive behavior (useful for dynamic environments).
4. supports multidimensional negotiations with interdependent dimensions.

The remaining five works are the ones that are closer to the above characteristics. (Sandholm, 1993) extended the original contract net protocol (CNP) in several ways, including the formalization of the bidding and awarding decision process and the possibility of dealing with clusters of tasks. To perform tasks the agents calculate the marginal cost based on their local criteria and the contractor selection is made based only on that cost. It is this price based mechanism that extends the original CNP to allow competitive and cooperative agents. Additionally, the cluster of tasks are negotiated as if they are a single proposal. The original CNP could only deal with a task at a time and, as such, ignoring the fact that, in same cases, the effort of executing a task depends on performing other tasks. The protocol is verified by the TRACONET (TRAnsportation COoperation NET) system, where dispatch centers of different companies cooperate automatically in vehicle routing.

This protocol is interesting but when compared with our proposal (GQN) it lacks some characteristics that, in our opinion, make it less suitable for other application domains: it is not multidimensional and does not support interdependencies between dimensions, there are no counterproposals and the agents do not have any adaptive strategies, i.e., during proposal formulation the agents propose, in one shot, a lower value than the cost received in the announcement. Additionally, the agents need to have full knowledge to present a proposal.

Another extension of the original CNP is presented by (Aknine *et al.*, 2004). Like in the previous work it allows competitive and cooperative agents supporting a single dimension only. However,

the choice is based on the time to perform a task and not in its marginal cost. According to the authors the protocol has the following advantages:

1. it enables M-N negotiations, i.e. a contractor agent can manage concurrently several negotiation processes with *m* manager agents, and a manager agent can manage concurrently several negotiation processes with *n* contractor agents.
2. it is more efficient in time and it is fault tolerant.

Regarding the first advantage, they support several concurrent 1-M negotiations through the inclusion of two new phases in the original CNP: *PreBidding* and *PreAssignment*. In our opinion and when compared with our proposal, this work has the same limitations than the previous one proposed by (Sandholm, 1993).

In three papers, (Lopes *et al.*, 2005, 2004, 2002) presents a multilateral, one-to-many, single or repeated round protocol that supports mixed environments. The agents are BDI possessing a library of negotiation strategies and tactics that are selected based on experience (the authors do not state how this selection is made). The model supports concession and problem resolution strategies, based on real world negotiations. These strategies are fixed and do not support adaptive characteristics. The negotiation tactics are used in two different moments: (1) to specify the proposal to submit in the beginning of the negotiation (in our proposal the agents do that according to their own private preferences); (2) to present a counter-proposal (in our proposal we use q-learning to learn the best counter-proposal to present).)

The agents are able to make a critic to a received proposal (a kind of feedback) regarding the attribute priority. They also support hard constraints, i.e., maximum attribute values that will not be relaxed and soft constraints, i.e., minimum acceptable attribute values that can be relaxed.

According to the authors, this protocol allows to dynamically define the negotiation problem and its representation, i.e., what is the set of negotiation attributes and how to add and remove attributes dynamically. In our proposal, the negotiation problem and its representation is dependent on the system designer.

It is important to point out that the authors performed experimentations using only two agents, Seller and Buyer, and one attribute only (price) in the negotiation object. This way, the reader is not able to see the full potential of the protocol, since the experimentations do not allow to validate all the characteristics conceptualized in the protocol.

Finally, when compared with our work we identify the following advantages of our proposal:

1. provides qualitative feedback over the values of all the attributes of the negotiation object (and not only about attribute priority.
2. besides supporting the definition of preferences over the attribute values the GQN supports interdependent dimensions.
3. the agents have adaptive strategies.
4. the protocol supports agents with full or partial knowledge, making it more suitable for distributed environments.

(Jain & Deshmukh, 2009) take advantage of fuzzy logic and developed an hybrid negotiation mechanism that combines competitive and cooperative negotiation. It is a bilateral, one-to-many, multidimensional protocol supporting interdependent dimensions and requiring agents with full knowledge to present proposal. The agents learn and make decisions on when to negotiate, with whom (using reinforcement learning) and how (using case-based learning) to negotiate based on the past negotiation experiences, current activities and predictions. According to the authors the

goal is not to get the optimal solution but the results are Pareto optimal, although the agents reveal the minimum information possible about their preferences and restrictions. The protocol is exemplified in a supply chain management scenario but can be used in other domains, specially in negotiations related to supply contracts for flexible production networks. We believe that GQN (our proposal) has characteristics that make it more suitable to a larger number of heterogeneous application domains, since we support more agents with heterogeneous behavior.

The last work we present here is a Representative-based Protocol developed by (Katsuhide Fujita & Klein, 2010) inspired on the parliamentary systems of England, Canada, Australia and Japan (amongst other countries). This work is an evolution from the Bidding-based Negotiation protocol presented earlier by the same authors. It is a multilateral, many-to-many, multi-round protocol supporting multiple interdependent dimensions. One difference regarding our protocol is that it uses non-linear utility functions instead of a weighted-sum function. According to the authors, this is an advantage of their protocol. To select the Representative agents (the ones that will participate in the negotiation) they use a mediator. The protocol supports competitive and cooperative agents and tries to maximize the Social Welfare. For each agent a *revealed area* is defined that represents the agent utility space that can be revealed to others. It is this utility space that defines a more competitive or more cooperative behavior of each agent. The agents tend to not reveal their information, however, they have an incentive to do it. The agent utility function is defined in terms of restrictions between attributes and, typically, each agent has its set of restrictions. The agents do not share its utility and use breadth-first search with branch cutting to find solutions and present proposals. Finally, in this protocol the agents' strategy is not adaptive and it is required to have full knowledge to present proposals making it less suitable to distributed and dynamic environments.

3.6 Agent Oriented Software Engineering

Agent technology in the context of software engineering has received a lot of attention during the last few years. Agent technology has been very successful from the scientific point of view as a metaphor for decentralized computation. From the commercial point of view we start to see some real-world agents and multi-agents systems applications. For example, the Distributed Computing with the Digipede Network[9] that uses agents to make distributed computing a reality, and the Infomobility Services application from the IST IMAGE project (Moraitis *et al.*, 2003b)

For the agent technology to succeed in the real world, an *Agent Oriented Software Engineering (AOSE)* is needed. According to (Henderson-Sellers & Giorgini, 2005) an AOSE is "*A methodological approach for the development of software oriented or based on agents*".

In this section we will present some AOSE methodologies and extensions to existing methodologies, that helped us preparing and proposing a new methodology called *PORTO* (see Chapter 5).

Table 3.3 presents a summary of the ones we refer in this section. It is not our intention to provide a full and detailed review of the state of the art regarding this subject. For more detailed information on the above methodologies and others that we did not mentioned here, we recommend the reading of (Bergenti *et al.*, 2004) and (Henderson-Sellers & Giorgini, 2005).

Table 3.3 has two parts. Table 3.3 a) lists six methodologies that we found relevant. As it is possible to see, all methodologies support the *Analysis* and *Design* phases, but not the *Test* or

[9] http://www.digipede.net

Table 3.3 Some AOSE Methodologies and Extensions

Authors	Name	Anal. Design	Tests	Implem.	Extension
Table 3.3 part a)					
(Caire *et al.*, 2001)	Message/UML	Yes	Yes	No	No
(Wood & DeLoach, 2001)	MaSE	Yes	No	Yes	No
(Padgham & Winikoff, 2002)	Prometheus	Yes	Yes	Yes	No
(Cossentino & Potts, 2002)	Passi	Yes	Yes	Yes	No
(Zambonelli *et al.*, 2003)	Gaia	Yes	No	No	No
(Bresciani *et al.*, 2004)	Tropos	Yes	Yes	Yes	No
Table 3.3 part b)					
(Moraitis *et al.*, 2003a)	Gaia based	Yes	No	Yes (Jade)	Yes
(Cossentino *et al.*, 2003)	Passi based	Yes	Yes	Yes	Yes (patterns)
(Cernuzzi & Zambonelli, 2004)	Gaia based	Yes	No	No	Yes (AUML)
(García-Ojeda *et al.*, 2005)	Gaia based	Yes	No	No	Yes (AUML)
(Moraitis & Spanoudakis, 2006)	Gaia2JADE	Yes	No	Yes (Jade)	Yes
(Castro & Oliveira, 2008)	Gaia based	Yes	No	Yes (Jade)	Yes (UML20)
(Passos *et al.*, 2011)	Gaia based	Yes	No	No	Yes (SOA)
(Silva *et al.*, 2012)	Gaia based	Yes	No	Yes (AgServ)	Yes

Implementation phases. For example, *GAIA* (Zambonelli *et al.*, 2003) that does not support either and *MaSE* (Wood & DeLoach, 2001) that does not support the test phase. Although not in the table, only three of them support requirements analysis, i.e., *Prometheus* (Padgham & Winikoff, 2002), *Passi* (Cossentino & Potts, 2002) and *Tropos* (Bresciani *et al.*, 2004).

Table 3.3 b) lists eight extensions to existing methodologies. One of them (Cossentino *et al.*, 2003) extends *Passi* by including patterns in all phases. The idea is to cut down the time and cost of developing multi-agent systems (MAS). According to the authors, patterns can be extremely successful with MAS (even more than with object oriented systems) because the great encapsulation of agents allows an easier identification and disposition of reusable parts.

Regarding the extensions based on *GAIA*, (Cernuzzi & Zambonelli, 2004) integrate *Agent UML (AUML)*[10] to improve modeling of open *MAS*, specifically, replacing the protocol model of GAIA with the Agents Interaction Protocol (*AIP*). (García-Ojeda *et al.*, 2005) propose the use of (AUML) as a notation to be used during the analysis and design phase, instead of the more informal notation adopted by *GAIA*. Specifically, and amongst other things, the authors propose a *AUML* representation of the organizational structure of the MAS.

Another extension that also proposes to replace the informal notation of *GAIA*, is our previous work (Castro & Oliveira, 2008). Instead of using *AUML* we propose the use of *UML 2.0*[11], since this notation is widely used in the industry and supported by several tools. Besides replacing the notation we also give some insights on how to use JADE (Bellifemine *et al.*, 2004) as the framework that could be used in the implementation phase of *GAIA*.

The use of JADE as an extension to the *GAIA* methodology is very popular. *Gaia2JADE* (Moraitis & Spanoudakis, 2006) enhances the Software Process Engineering Metamodel proposed by the Object Management Group[12], adding the JADE development phase, proposing a process

[10] http://www.auml.org

[11] http://www.uml.org/

[12] http://www.omg.org

that covers the full software development cycle. Another example is the work of (Moraitis *et al.*, 2003a).

Finally, two very interesting extensions have been proposed by (Passos *et al.*, 2011) and (Silva *et al.*, 2012). Regarding the first, *Passos et. al.* analyze the adequacy of traditional *AOSE* to be used to create adequate *MAS* to the transportation domain. Starting from *GAIA* and from the extension proposed by (Castro & Oliveira, 2008), they propose a novel methodology where the concept of services is considered as peer of agents, ambient and processes. As such, they apply the concepts of *Service Oriented Architecture (SOA)* to it. They have used the approach to analyze, design and implement a planner system for generating multimodal trips (various types of transportation).

Regarding the second, *Silva et. al.* uses an adapted version of *GAIA* to design an abstract generic system meta-model for a multi-robot application, which can be used as a basis for the design of these systems, avoiding or shortening repetitive tasks common to most systems. Like with the first work, they also enhance some of the extensions proposed in (Castro & Oliveira, 2008). Based on the proposed *Generic Robotic Agent Meta-Model (GRAMM)*, two distinct models for two different applications were derived, showing the versatility and adaptability of the meta-model.

In Chapter 5 we propose a *GAIA* based methodology called *PORTO* that includes all phases, making it one of the most complete *AOSE* methodologies.

3.7 Chapter Summary

The main goal of this chapter was to present a comprehensive analysis of the work that has been done regarding disruption management in airline operations control. For that, we have proposed a classification scheme that was used to classify and compare the existing work. For ease of reading we have included tables that summarize the most relevant information of the comparative analysis that was made.

Additionally, we have presented some research work regarding automated negotiation and agent-oriented software engineering (AOSE). This information will allow the reader to better understand our proposal for a new AOSE methodology (Chapter 5) and for a new automated negotiation protocol (Chapter 6).

In the next chapter we present the *Airline Operations Control Problem*. It is an important chapter, since it will allow the reader to understand the problem we want to solve as well as its characteristics.

Chapter 4
The Airline Operations Control Problem

Abstract Chapter 3 was mainly about the state of the art in airline operations' disruption management and related fields. We have additionally proposed a possible classification encompassing the existing work within this field. This chapter is about the main application domain we are interested in and the knowledge it entails. The information provided here is the result of the observation and interviews we have done at TAP Portugal[1] Airline Operations Control Center (AOCC), complemented with information from related literature. As we stated in Chapter 1 and considering the line of research we chose to follow, it is important to have a rigorous description of the real application domain scenario as well as the problem to be solved. This chapter is a source of empirical knowledge, that will help the reader to understand both. We will introduce the AOCC organization and how it works, the type of problems that the AOCC human operators have to solve as well as the most common solutions and methods used therefore. Some real statistical data from TAP Portugal will be presented as well as the main costs involved in possible solutions.

4.1 Introduction

One of the most important tasks of an airline is to control the operation plan, i.e., make sure that the flights are executed according to the scheduled plan. It is inconsequential to produce optimal, or near-optimal, flight schedules if, later on, during execution, disruptions regularly cause significative deviations to the original schedule. Unfortunately, the majority of the disruptions are difficult to predict (for example, those caused by meteorological conditions or aircraft malfunctions). Airline companies developed a set of operation control mechanisms to monitor the flights (and crew members) to check for compliance. During this monitoring phase, several problems may appear related to aircraft, crew members and passengers (Clausen *et al.*, 2005).

According to (Kohl *et al.*, 2004), disruption management is the process of solving these problems. To be able to manage disruptions, airline companies have an entity called Airline Operations Control Center (AOCC). This entity is composed of specialized human teams that work under the control of an operations supervisor. Although each team has a specific goal (for example, the crew team is responsible for having the right crew in each flight), they all contribute to the more general objective of minimizing the effects of disruption in the airline operational plan.

In this chapter we introduce the airline operations control problem - AOCP (also known as airline disruption management problem). To contextualize the problem at hands, we start by briefly introducing the AOCP's preceding problem known as the Airline Scheduling Problem (ASP) in Section 4.2. Then, in Section 4.3, we explain what an airline operational control center (AOCC) is and we present some typical AOCC organizations. The AOCC information sources are presented in Section 4.4. The typical problems, the current disruption management process as well as the main costs involved are also introduced in Sections 4.5, 4.6 and 4.7, respectively. A brief problem statement is presented in Section 4.8 and we end with a chapter summary in Section 4.9.

[1] `http://www.flytap.pt`

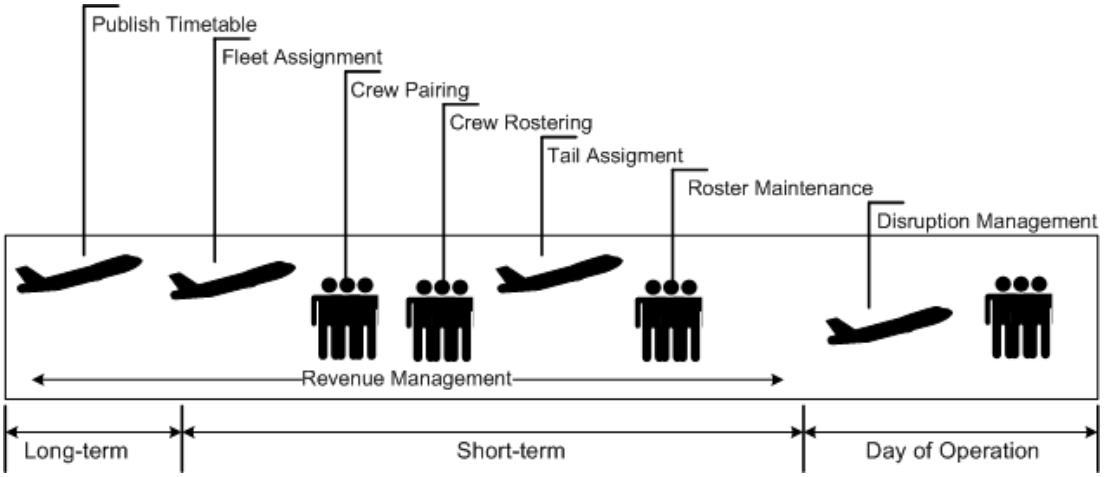


Fig. 4.1 The airline scheduling process

4.2 Airline Scheduling Problem

According to (Kohl *et al.*, 2004) the scheduling process of an airline company is composed by the long and short-term phases presented in Figure 4.1. The scheduling process has three main dimensions or views:

1. Passenger View
2. Aircraft View
3. Crew View

The first one represents the seats available to be sold to the airline customers. The other two views, represent resources that will be allocated.

Everything starts with *publishing the flights timetable* for a specific period of time (usually six months). After publishing the timetable, the *revenue management* phase starts. Here the goal is to maximize the revenue obtained by selling tickets. At the same time, the scheduling of the two most important resources starts: aircraft and crew.

Regarding the aircraft, the first step is *fleet assignment*. Here, the goal is to assign the aircraft type or aircraft fleet that will perform the flights. It is an important step because the aircraft type/fleet will define the number of available seats in each flight. Nearer to the day of operations, the assignment of the specific aircraft to each flight is performed. This step is known as *tail assignment*.

After the fleet assignment step, it is possible to start to schedule the crew. The first step is the *crew pairing*. The goal is to define the crew duty periods (pairings) that will be necessary to cover all the flights of the airline for a specific period of time (typical one month). Having the pairings completed, it is possible to start the *crew rostering* step that is, assign crew members to the pairings. The output of this step is an individual crew roster that is distributed or published in the crew web portal. Finally and until the day of operations, it is necessary to change/update the crew roster (*roster maintenance*), to include any changes that might appear after publishing the roster.

The airline scheduling problem (ASP) is composed of all the previous phases and steps and ends some hours or days (depends on the airline's policy) before the day of operation. The global objective of the ASP is to maximize the airline operating profit. There are several tools available to

help airline companies to optimize several phases of the ASP. The study of those tools is out of the scope of our work. However, Tobias Grosche has an excellent book about this subject (Grosche, 2009). We invite the interested reader to read the book, specially Section 2.1 to Section 2.4.

4.3 AOCC Organization

The airline operations control problem (AOCP) starts where the airline scheduling problem stops. If everything goes as planned the airline just needs to monitor the execution of the plan. Unfortunately, several unexpected events usually appear during this phase that can disrupt the plan. To monitor those events and solve the problems arising from them, it is necessary to define and follow a disruption management process. Airline companies have an entity called Airline Operations Control Center (AOCC) that is responsible for the disruption management process. The role or support functions more common in an AOCC, according to (Kohl *et al.*, 2004) and (Castro, 2008), are the following:

- *Flight Dispatch:* Prepares the flight plans and the request of new flight slots to the Air Traffic Control (ATC) entities (FAA in North America and EUROCONTROL in Europe, for example).
- *Aircraft Control:* Manages the resource *aircraft*. It is the central coordination role in the operational control. In a disruptive situation, tries to minimize the delays by changing aircraft and rerouting or joining flights, among other actions. Usually, uses some kind of computer system to monitor the operation that, in some cases, might include some decision support tools. Much more common is the use of *rules-of-thumb* based on work experience (a kind of hidden knowledge).
- *Crew Control:* Manages the resource *crew*. Monitors the crew check-in and check-out, updates and changes the crew roster according to the disruptions that might appear during the operation. As in the previous role, it uses some kind of system with or without decision support tools. Experience and use of *rules-of-thumb* are still the most common decision aiding tools. Using reserve crew and exchange crew members from other flights, are among the possible actions used to solve crew problems.
- *Maintenance Services:* Responsible for the unplanned maintenance services and for short-term maintenance scheduling. Changes on aircraft rotation may impact the short-term maintenance (maintenance cannot be performed at all stations).
- *Passenger Services:* Decisions taken by the AOCC will have an impact on the passengers. The responsibility of this role is to consider and minimize the impact of the decisions on passengers, trying to minimize the passenger trip time. Part of this role is performed on the airports and for bigger companies it is part of the HCC organization.

In Section 2.5 we have provided some theoretical information about the structure of the organizations relating it with our previous study (Castro, 2008) about the ideal structure for an AOCC. In this section we are going to present the three main types of AOCC organizations (Castro, 2008), that are commonly seen in airlines:

- *Decision Center:* The aircraft controllers share the same physical space. The other individual roles or support functions (crew control, maintenance service, etc.) are in a different physical space. In this case the AOCC has autonomy to take decisions regarding the flights (delay, cancel, join flights, etc.) but it needs the cooperation of the other support functions, because they do

not depend hierarchically from the AOCC. In Figure 4.2 we show an example of this kind of organization. As it is possible to see, the role of the *AOCC Supervisor* is at the same level as the other roles. In this type of *Collective Organization* all roles need to cooperate to achieve the common goal.

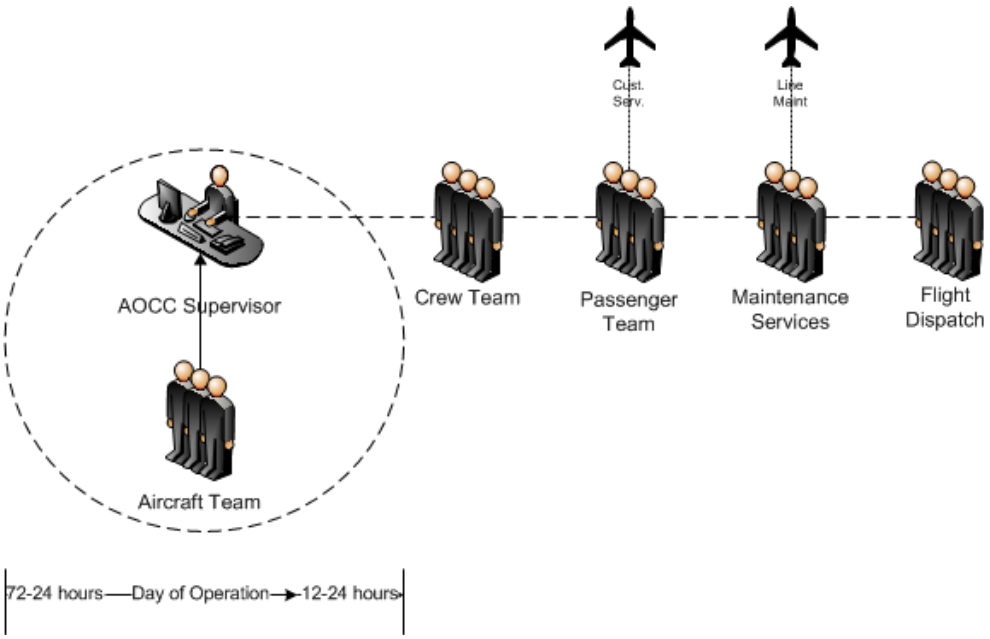


Fig. 4.2 AOCC Decision Center

- *Integrated Center:* All roles share the same physical space and are hierarchically dependent of the *AOCC Supervisor*. For small companies we have a *Simple Hierarchy Organization*. For bigger companies we have a *Multidimensional Hierarchy Organization*. Figure 4.3 shows an example of this kind of AOCC organization. The main advantage is that the final decision is taken by one person only (although at the lower levels, it is common and sometimes necessary, some cooperation).
- *Hub Control Center (HCC):* Most of the roles are physically separated at the airports where the airline companies operate a hub. In this case, if the aircraft controller role stays physically outside the hub we have an organization called *Decision Center with a hub* (see Figure 4.4). If both the aircraft controller's and crew controller's roles are physically outside the hub we have an organization called *Integrated Center with a hub* (see Figure 4.5). The main advantage of this kind of organization is to have the roles related to airport operations (customer service, catering, cleaning, passengers transfer, etc.) physically closer to the operations themselves.

The organization adopted depends on several factors like airline size, airline network type (for example, hub-and-spoke[2]) and geographic distribution of the operation, as well as, tradition and/or company culture. In Figure 4.3 we present the organization of a typical *Integrated Operational Control Center*. It is important to point out the role of the supervisor, which gives this organization its hierarchical characteristic and, also, the operation's time frame, which marks the responsibil-

[2] A system of air transportation in which local airports offer air transportation to a central airport where long-distance flights are available

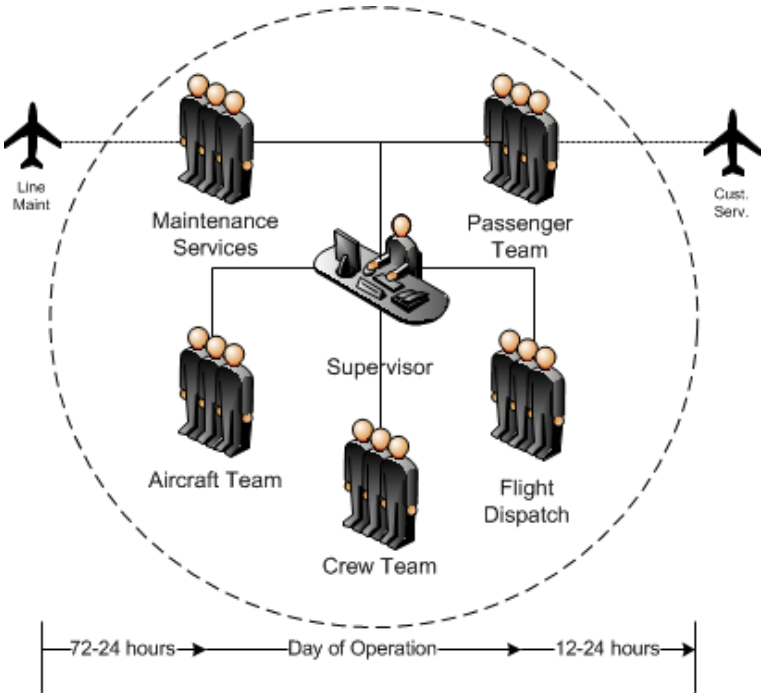


Fig. 4.3 Integrated AOCC

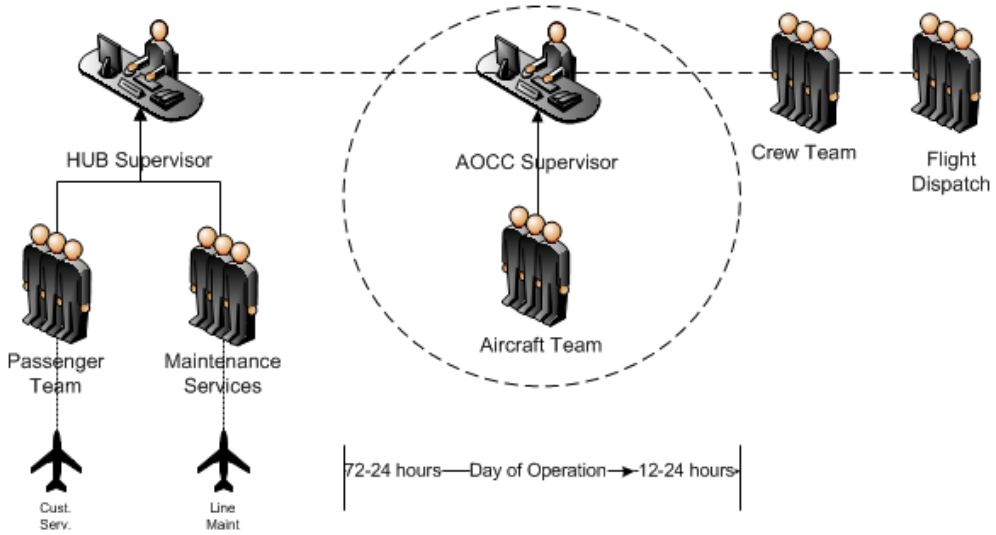


Fig. 4.4 AOCC Decision Center with a HUB

ity boundaries of the AOCC. This operation's time frame is different from airline to airline but, usually, ranges between the 72 or 24 hours previous to the operation, to the 12 or 24 hours after.

4.4 AOCC Information Sources

The AOCC needs to have access to several information sources to be able to perform its role. In this section we will present the main information sources needed, according to the interviews we

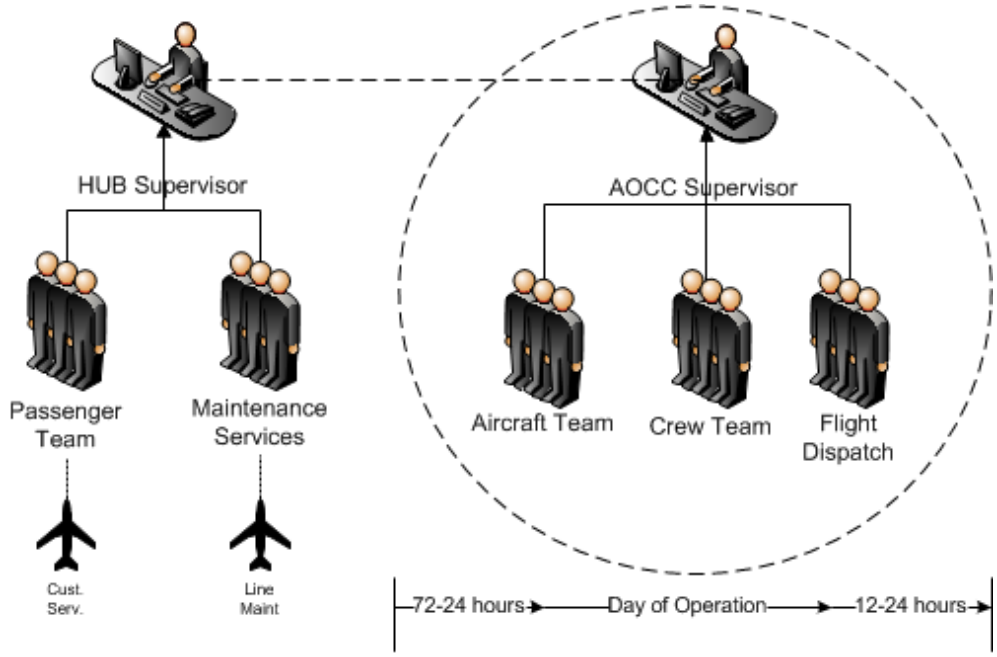


Fig. 4.5 Integrated AOCC with a HUB

have performed as well as our own research. We will also include, when possible, some of the systems and tools that can provide that information. Table 4.1 summarizes these findings.

4.5 Typical Problems

In the previous section we presented typical AOCC organizations as well as the roles pertaining to those organizations. Now, it is important to understand the typical problems that appear during the execution of the airline's operations. As a result of our empirical observations of a real AOCC, and the report of (Kohl & Karisch, 2004), we found a set of typical problems presented in Figure 4.6. In this diagram we have also included the impact that each problem might have on flight arrival or departure delays as well as the relation existing between them. The diagram also shows that the problems may propagate due to the relation between them and generate new problems on different flights. This propagation characteristic creates a more difficult problem to solve optimally in a real time dynamic environment, like the one we have on the AOCC.

Before proceeding it is important to note that in this context, the word *problem* has a broader meaning. Depending on the point of view, a specific event might be a cause of something or a problem that needs to be solve. For example, the event *Loading Delay* from the point of view of the ground handling agent[3] is a problem that needs to be solved, i.e., they need to take specific actions so that the aircraft loading is not delayed. From the point of view of the AOCC this is an event that might cause a flight departure delay, making the latter a problem to be solved. There are other events that, from the point of view of the AOCC are, simultaneously, events and problems they need to solve. For example, *Crew Delay* is an event that might cause a Flight Departure Delay

[3] Ground handling addresses the many service requirements of an airliner between the time a flight arrives at a terminal gate and the time it departs for its next flight.

Table 4.1 AOCC Information Sources

Source	Description
Flight and Aircraft Movement	This is one of the most important sources of information on the AOCC. Through this source the AOCC knows when the flight/aircraft departs and arrives to its destination. It has information about scheduled times as well as information about actual times and cancellations and/or aircraft changes. Additionally, it has information about the passengers of each flight and its flight's connections. This information source is updated through MVT messages and Datalink (ACARS) messages as well as by flight reports (reports filled and submitted by the flight crew). Some airline companies have their own system that collects the information received from these different sources. For example, TAP has the MCS (Movement Control System) that is part of Compass, a larger system that integrates many other information. There are also additional tools, like CDM (Collaborative Decision Making) that provide information about all flights (from the same company or not) in specific airports.
Aircraft Roster	This source provides the flights as well as maintenance activities scheduled for each aircraft. This helps the Aircraft Team as well as Maintenance services to take decisions.
Crew Roster	This source contains the schedule activities for each crew member of the company. Besides flights, it also includes days off, vacation, training and any other information that is relevant. It helps the Crew Team to take decisions.
Passenger Booking	Information about the itinerary of each passenger as well as about the available seats in each flight. It also provides information about the boarding status of each flight. It helps the Passenger Team in taking decisions and in finding alternative itineraries for each disrupted passenger (Definition 3.5).
ATC Slots	Information about available times to land or take-off at specific airports or to overfly specific waypoints. In Europe the main tool for that is the CFMU (Central Flow Management Unit).
Meteorological Information	Information about airport and en-route weather. Nowadays most of this information is available through the internet. However, other sources like the National Weather services are also available. The biggest airline companies tend to have their own meteorological department, providing this information internally.
Aircraft/Flight Costs	Ideally, the AOCC should have access to on-time and updated information related to: airport costs (approach and taxing taxes, for example), service costs (cleaning services, handling services, line maintenance, etc.), and average maintenance costs for the type of aircraft, ATC en-route charges and fuel consumption. Unfortunately, most of the airline companies do not provide this information to the AOCC on-time and updated.
Crew Costs	Like the previous one, the AOCC should have access to the average or real salary costs of the crew members, additional work hours and *perdiem* days to be paid, hotel costs and extra-crew travel costs. However, it has the same availability restrictions of the previous one.
Passenger Costs	Passenger airport meals, passenger hotel costs and passenger compensations. This information is important to the decisions that the Passenger Team has to take.

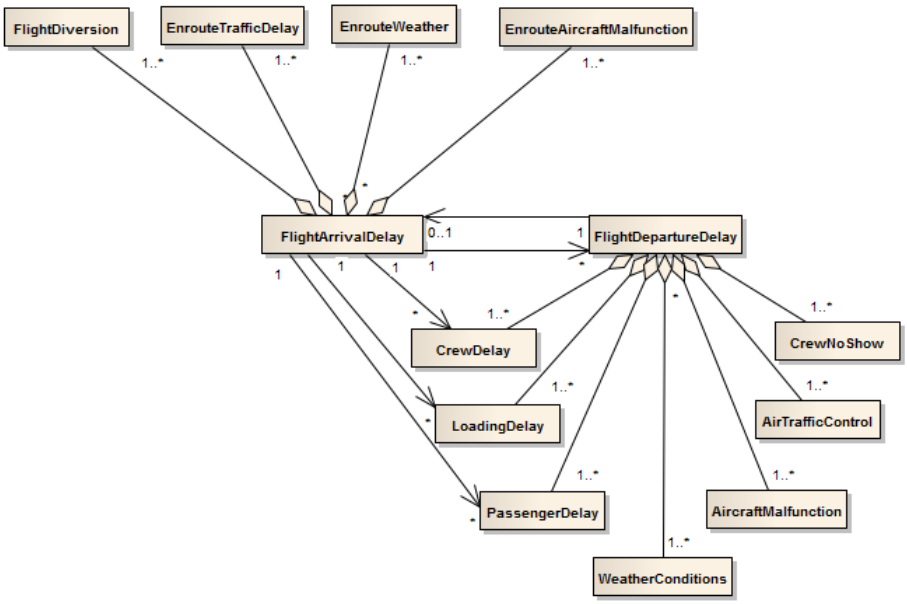


Fig. 4.6 Typical AOCC Problems and Relations

problem but, at the same time, it is a problem that the AOCC needs to solve by itself (for example, by using a reserve crew member to replace the delayed one).

Adopting the point of view of the AOCC in this section, we say that the main problems ($Prob_{aoc}$) are *Flight Departure Delays* (F_{dd}) and *Flight Arrival Delays* (F_{ad}), i.e.,

$$Prob_{aoc} = \begin{cases} F_{dd} & \text{, if etd>std} \\ F_{ad} & \text{, if eta>sta.} \end{cases} \tag{4.1}$$

As we can see in Figure 4.6 there is an obvious relation between *Flight Arrival Delays* and *Flight Departure Delays*. Most of the flights are performed by aircraft that are used in previous flights. If we have an arrival delay and the aircraft turnaround time[4] at the airport is not enough, then, if the AOCC does not find an alternative solution, we will also have a departure delay.

From the diagram we can also see that the main events that cause flight arrival delays (besides the delay on departure) are:

- En-route air traffic.
- En-route weather.
- En-route aircraft malfunction.
- Flight diversion

In these cases and to minimize the arrival delay a cooperation is necessary between the pilot, AOCC and ATC. As a side note, which will be pointed out latter in Section 7.8, the technological trilogy *Datalink* (a World Wide Aircraft Communication Network) plus *Electronic Flight Bag* (EFB) plus *Pilot Action* can have a very important proactive role in recovering arrival delays and, as such, also helping to minimize or avoid departure delays. Regarding departure delays, the main events are:

- Crew delays.

[4] The time during which the aircraft must remain parked at the airport gate, starting at flight arrival and ending at flight departure.

- Cargo/baggage loading delays.
- Passenger delays.
- Crew members that do not report for duty.
- Air traffic control restrictions.
- Aircraft malfunctions.
- Weather conditions (either at the departure airport or at the arrival airport).

As we stated before, TAP Portugal provided us with data regarding its operations. As with any airline, TAP records the reason for each flight delay following the *IATA*[5] numeric delay codes plus its own proprietary delay codes. In appendix D we present both tables and, as it is possible to see, it contains many codes and labels. These codes are useful to classify and study the delays *after* they happen. However, to be able to relate the delays and its causes according to the resource affected and use this information in a *proactive* way, trying to avoid or minimize the delays *before* they happen, we need to have much fewer codes or classification categories.

Using the information we got from the interviews performed to the AOCC human operators of TAP, we defined the *Flight/Aircraft* category of events according to Table 4.2 and the *Crew members* events according to Table 4.3.

Table 4.2 Flight/Aircraft events category

Event Category	Description
AIRP	Airport infrastructure causes: SEF, Stands, Airport Capacity, ULDs, RX machines, etc.
ATC	*En route* and destination ATC restrictions as well as en route and destination meteorological conditions.
COMM	Protection of passengers due to cancellation of another flight or passengers missing after check-in.
HAND	Problems boarding passengers and/or loading cargo, problems due to taxing/runway officer.
MAINT	Problems due to some kind of malfunction and/or maintenance related.
METEO	Adverse meteorological conditions at departure airport impairing landing or handling.
CREW	Missing crew members and other crew related problems.
ROT	Problems related to the rotation of the aircraft. For example, late arrival of the incoming aircraft.
SEC	Baggage identification, search and retrieve of baggage after boarding.
OTH	All other problems not related to any of the previous.

Using these categories we have classified the IATA and TAP delay codes accordingly (see appendix D) and we were able to statistical analyze one full year of TAP delays (from April 2009 to April 2010) and relate that information with the flight and crew events category of Tables 4.2 and 4.3. In Figure 4.7 we show the average of the flight delays by events category. As it is possible to see, the three main reasons for the delays are:

1. *ATC* reasons: on average 18% of the flight delays.
2. *HAND* reasons: on average 17% of the flight delays.
3. *ROT* reasons: on average 16% of the flight delays

[5] International Air Transport Association - http://www.iata.org

Table 4.3 Crew events category

Event Category	Description
SIGN	Crew member not reporting for duty at home base.
RULES	Crew member has exceeded any labor/law rules like duty time.
INDUTY	Crew member unavailable after sign-on. For instance, accident or illness during flight.
ROT	Crew member miss a flight due to the delay of a previous flight or other causes related to crew rotation.
METEO	Adverse meteorological conditions at departure airport impairing landing or handling.
OTH	Other reasons not included in the previous.

This information will be very helpful in understanding how and why the human operators solve some of the problems in the AOCC and to understand what we call the *Probabilistic Solution Action Tables* that we will present in the next section.

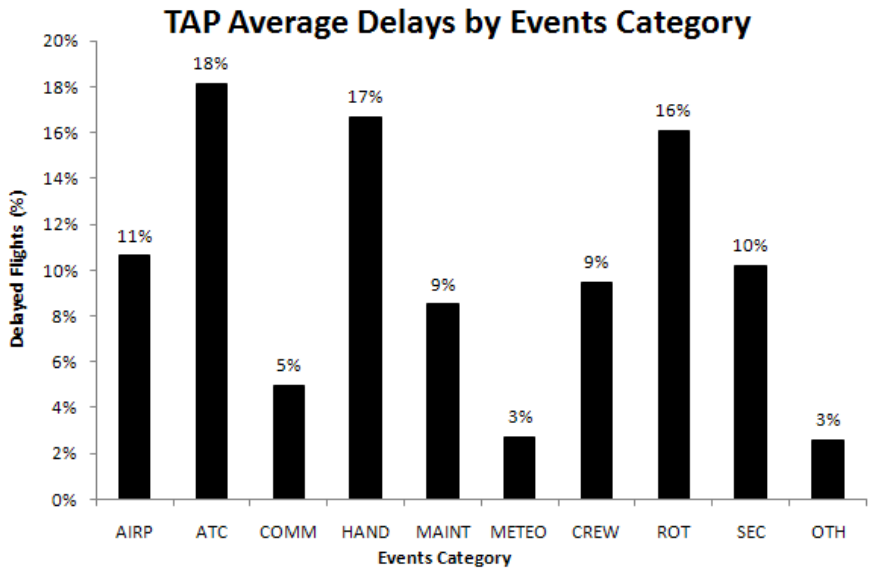


Fig. 4.7 TAP Delays by Events Category

4.6 Current Disruption Management Process

As we can see from the previous section, there are several events that may cause flight delays. AOCCs have a process to monitor the events and solve the problems, so that flight delays are minimized with minimum impact on passenger and, preferably, with the minimal operational cost.

In Figure 4.8 we present the current disruption management process in use at most of the airlines. This process has five steps:

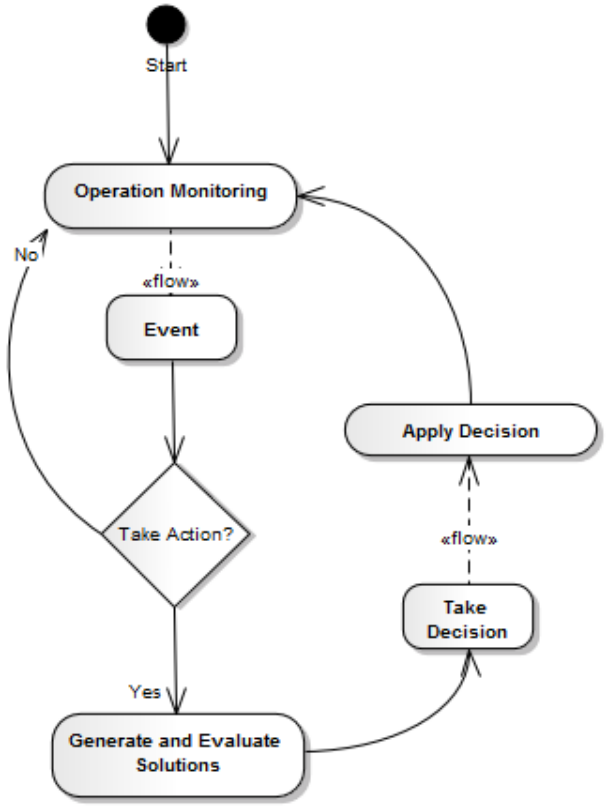


Fig. 4.8 AOCC Disruption Management Process

1. ***Operation Monitoring:*** In this step the flights are monitored to see if anything is not going according to the plan. The same happens in relation to crew members, passenger check-in and boarding, cargo and baggage loading, etc.
2. ***Take Action:*** If an event happens, like for example, a crew member is delayed or an aircraft malfunction, a quick assessment is performed to see if an action is required. If not, the monitoring continues. If an action is necessary then we have a problem that needs to be solved.
3. ***Generate and Evaluate Candidate Solutions:*** Having all the information regarding the problem the AOCC needs to find and evaluate the candidate solutions. Usually, a *sequential approach* is adopted when generating the solutions. First, the aircraft problem is solved. Then, the crew problem and finally, the passengers. It is understandable that the AOCC adopts this approach. Without good computer tools, it is difficult to take care of the problem, considering the three dimensions (aircraft, crew and passengers) simultaneously. Although there are many costs involved in this process, we found that the AOCC relies heavily on the experience of their controllers and in some *rules-of-thumb* (a kind of hidden knowledge) that exist on the AOCC.
4. ***Take Decision:*** Having the candidate solutions, a decision needs to be taken.
5. ***Apply Decision:*** After the decision the final solution needs to be applied to the environment, that is, the operational plan needs to be updated accordingly.

In our opinion, this process can benefit to a great extent from an intelligent agent based approach to the problem, as we will explain in Chapter 7.

During the interviews we have performed to TAP AOC human operators we found that in step 3 (*Generate and evaluate candidate solutions*) the human operators have some *rules-of-thumb* they use to solve the problems and generate candidate solutions. These rules are different according

to the role or expertise of the human operator. For example, the *Aircraft team* responsible for controlling the flights and aircraft in the AOC, when faced with an event that might cause a *flight departure delay* they follow the *General Process of Aircraft Problem Resolution*:

1. *Exchange (or swap) the aircraft* of the disrupted flight (Definition 3.4) with an available aircraft or with an aircraft assigned to a flight with a later STD. That aircraft needs to be at the same airport and with seat capacity to take all the passengers of the disrupted flight. It means that the aircraft and the crew members of that aircraft and flight need to be available.
2. *Rerouting*, i.e., fly to the destination using a different route. It makes sense when the delay reason is related to slots[6]. It implies ATC authorization because the flight plan will be changed.
3. *Join Flights*, i.e., try to have an already assigned aircraft to a different flight to also perform the disrupted flight. It implies crew members availability, ATC authorization and enough available passenger seats to also take the passengers from the disrupted flight.
4. *Delay* the flight. This is the *default* solution.
5. *ACMI*, i.e., the leasing of an aircraft including the crew, maintenance services and insurance. It requires authorization from the airline administration and it is usually done when a crew strike occurs or when it is not possible to send the passengers in other flights.
6. *Cancel* the flight.

From this *General Process of Aircraft Problem Resolution* it is possible to identify the *Actions* taken by the human operators that lead to candidate solutions, as well as a preliminary format or definition of those candidate solutions (at this moment we are adopting a very informal way of defining it). Table 4.4 summarizes this findings. The first thing to notice is that the general format

Table 4.4 Actions and candidate solutions for Aircraft Problems

Action	Preliminary Candidate Solution Definition
EXCHANGE	$Sol = \{\langle airc_o, flt_d\rangle, \langle airc_d, flt_o\rangle\}, flt = \langle Leg_1, ..., Leg_n\rangle$
REROUTING	$Sol = \{\langle airc_d, flt_d\rangle\}$ (flight plan change)
JOIN	$Sol = \{\langle airc_o, flt_o + flt_d\rangle, \langle airc_d, --\rangle\}$
DELAY	$Sol = \{\langle airc_d, flt_d\rangle\}, etd = std + delay$
ACMI	$Sol = \{\langle airc_{acmi}, flt_d\rangle\}$
CANCEL	$Sol = \{\langle airc_d, --\rangle\}$

for a solution is a set of one or more pairs of aircraft/flights, i.e., $\langle airc_1, flt_1, \ldots, airc_n, flt_n\rangle$. For example, considering the first action *Exchange*, there are two pairs: one for the disrupted flight $\langle airc_d, flt_d\rangle$ and another for a different flight $\langle airc_o, flt_o\rangle$. So, by exchanging aircraft the solution will be the set $\{\langle airc_o, flt_d\rangle, \langle airc_d, flt_o\rangle\}$.

As described above, in the *General Process of Aircraft Problem Resolution* we have six actions, each one leading to a different solution. When faced with an event that leads to an aircraft problem which action should be taken? Should we follow the order of the general process and use the first one that works or do the aircraft team follows some other *rule-of-thumb*? From the interviews we found that the use of an action has some relation to the event that caused the problem. For example, typically, if the event is an aircraft malfunction the most common action to use is to exchange aircraft. Using this information plus the *Flight/Aircraft events category* table 4.2 we have defined the *Aircraft Team Probabilistic Solution Action* table 4.5. For example, using the Table 4.5 we can

[6] The permission to land and take-off at a specific date and time

Table 4.5 Aircraft Team Probabilistic Solution Action

Action	AIRP	ATC	COMM	HAND	MAINT	METEO	CREW	ROT	SEC	OTH	Avg
EXCHANGE	0%	0%	0%	0%	80%	0%	0%	80%	0%	60%	22%
REROUTING	0%	50%	0%	0%	0%	0%	0%	0%	0%	7%	6%
JOIN	0%	0%	0%	0%	2%	0%	0%	0%	0%	4%	1%
DELAY	95%	45%	90%	98%	15%	98%	95%	19%	100%	25%	68%
ACMI	0%	0%	0%	0%	1%	0%	0%	0%	0%	1%	0%
CANCEL	5%	5%	10%	2%	2%	2%	5%	1%	0%	3%	4%

see that if the event that caused a flight delay is related to maintenance (*MAINT*), in 80% of the problems the action *EXCHANGE* will be the best one to use. The *DELAY* one will be the best to use in 15% of the problems and *JOIN*, *CANCEL* and *ACMI* in 2%, 2% and 1% of the problems, respectively. This information is very important, not only for *manual problem solving* processes but, also, to implement any kind of computerized system. As we will see in Chapter 7 we will use this table to provide the initial information to the learning mechanism of the software agents.

Regarding the *Crew Team* responsible for controlling the crew members in the AOC, they also follow a rather more complex *General Process of Crew Problem Resolution*. This process includes the following actions:

- *Use Reserve at Airport*, i.e., to use a reserve crew member available at the airport of the disrupted crew (Definition 3.3) flight.
- *Use Nearest Reserve at Home*, i.e., to use a reserve crew that lives nearest the airport of the disrupted crew flight.
- *Exchange with Crew from another Flight*, i.e., to exchange the disrupted crew with a crew member assigned to a different flight with a later STD.
- *Use crew with Free Time*, i.e., if available, use a crew member that does not have a flight or activity assigned.
- *Use Day Off Crew*, i.e., use a crew member that has a day off. Usually, requires permission from the crew member.
- *Use Crew on Vacation*, i.e., use a crew member that is on vacation. Requires permission from the crew member.
- *Propose an Aircraft Change*. If the only replacement crews available do not have qualifications to fly the aircraft type assigned to the flight of the disrupted crew member, then it might be possible to change the aircraft type. In this case, the crew team proposes this change to the aircraft team. Usually, it is used when the disrupted crew member is a pilot, because they have more restrictions regarding flying different aircraft types.
- *Proceed without Crew*, i.e., if possible by law the flight can proceed without the disrupted crew member.
- *Cancel the Flight*, i.e., if there is no replacement available and the law does not allow to proceed without the disrupted crew member, then to cancel the flight is an option that the crew team will propose to the aircraft team.
- *Accept Delay*, i.e., wait for the disrupted crew member and delay the flight accordingly.

In the case of the *Actions and Candidate Solutions for Crew Problems* the general format for a solution is a set of one or more pairs of crew member/flight, i.e., $\langle crew_1, flt_1, \ldots, crew_n, flt_n \rangle$. For example, considering the action *Exchange with Crew from another Flight*, there are two pairs:

one for the disrupted crew $\langle crew_d, flt_d \rangle$ and another for a crew on a different flight $\langle crew_o, flt_o \rangle$. Exchanging crew members the solution will be the set $\{\langle crew_o, flt_d \rangle, \langle crew_d, flt_o \rangle\}$.

As we have done for the aircraft team, using this information plus the *Crew events category* table 4.3 we have defined the *Crew Team Probabilistic Solution Action* table 4.6. As an example

Table 4.6 Crew Team Probabilistic Solution Action

Action	SIGN	RULES	INDUTY	ROT	METEO	Avg
Use reserve at airport	10%	5%	0%	20%	0%	7%
Use nearest reserve at home	40%	20%	0%	40%	0%	20%
Exchange with crew another flt	10%	10%	0%	10%	0%	6%
Use crew with free time	10%	10%	0%	5%	0%	5%
Use day off crew	10%	10%	0%	10%	0%	6%
Use crew on vacation	5%	10%	0%	5%	0%	4%
Propose aircraft change	1%	0%	0%	0%	0%	0%
Proceed without crew	2%	5%	90%	5%	0%	20%
Cancel flight	2%	10%	10%	0%	2%	5%
Accept delay	10%	20%	0%	5%	98%	27%

and using the Table 4.6 we can see that if the event that caused a *disrupted crew member* is related to a crew member problem after sign-on (*INDUTY*), in 90% of the cases the action *Proceed without crew* is the best one to use. The *Cancel Flight* action will be used on the rest 10% of the cases.

Finally, we need to analyze the role of the *Passenger Team*. The events and problems that this team has to deal with, are a little bit different from the ones in the aircraft and crew team. From a certain point of view, the passenger problems are a consequence of the actions taken and the solutions found to solve the aircraft and crew problem. If a flight is not delayed then we do not have *disrupted passengers* (Definition 3.5). Nevertheless, when faced with one or more disrupted passengers, the passenger team follows the *General Process of Passenger Problem Resolution*:

1. *Change Passenger to Another Flight in the Same Airline*. This is the preferred solution. It is less expensive to the company to find an alternate flight for the passenger if that flight is performed by the same airline company. The business class passengers and/or other important passenger (VIP's, Frequent Passengers, etc.) have priority over the others.
2. *Change Passenger to Another Flight on Other Airline*. If there is no flight in the same company, then the alternative is to send the passenger in a flight that is performed by a different airline. Like in the previous action, special passengers have priority.
3. *Keep passenger Delayed Flight*. This is the default solution if nothing else works.

Although in this case the action to take is not related to the event that caused the problem, the passenger team also has a kind of *Passenger Team Probabilistic Solution Action* as presented in Table 4.7.

Table 4.7 Passenger Team Probabilistic Solution Action

Action	Avg
Change Pax Another Flight Same Company	70%
Change Pax Another Flight Other Company	10%
Keep Passenger in Delayed Flight	20%

A final word regarding the general format for a solution of a passenger problem. In this case, the solution is an ordered set of flights, starting at the airport of the disrupted flight and ending at the airport of the final destination of the passenger, i.e., $\{flt_1, \ldots, flt_n\}$.

To conclude this section we just need to explain the typical approach adopted by the AOCC operators to solve a Flight Delay problem. As we have stated in a previous section, a Flight Delay problem has three sub-parts or sub-problems to be solved. Two of them, aircraft and crew members, are resources assigned to the flight. The third part are the passengers of the flight. The typical approach used is the following:

1. First, the *Aircraft* sub-problem is solved, i.e., a flight to be performed needs an aircraft. So, the first thing to do is to make sure that there is an available aircraft at the airport of the delayed flight, preferably ready to departure at the STD of the flight and with enough seats available to take all the passengers of the disrupted flight.
2. Second, the *Crew* sub-problem is solved, i.e., besides an aircraft we need a crew for the flight to be performed. So, after solving the first part of the problem, it is necessary to have the right crew. This means to have a flight crew and cabin crew qualified to fly that specific type of aircraft, in the right number, at the airport of the disrupted flight and, preferably, on the aircraft before the STD.
3. Third, the *Passenger* sub-problem is solved, i.e., now that we have an aircraft and crew and an ETD, we can see if there are passengers that will miss any connection they have until they reach the final destination. In that case, a new itinerary needs to be found for all the disrupted passengers.

At first sight, this sequential approach makes sense and seems to be the ideal one to use. However, as we will show later, this approach tends to optimize the sub-problems that are solved first. We think that we can improve the results by using a computational approach that takes into consideration, simultaneously, all the sub-problems in order to reach a better integrated solution. We will show our proposed solution in Chapter 7 and the experimentation results in Chapter 8.

4.7 Main Costs Involved

In the step *Generate and Evaluate Solutions* of the disruption management process presented in Figure 4.8, we should consider the main costs involved in generating and choosing from candidate solutions. According to our empirical observations these are the main costs involved when generating and evaluating a solution for a specific disruption:

1. ***Flight/Aircraft Costs:*** airport costs (takeoff and landing taxes, parking and handling charges), service costs (cleaning services and line maintenance), maintenance costs for the type of aircraft, ATC en-route charges and fuel consumption (differs for each aircraft type).
2. ***Crew Costs:*** salary costs of the crew members (different according to the salary rank), additional work hours and *perdiem* days to be paid, hotel costs and extra-crew travel costs (happens when a on duty crew member travels as a passenger).
3. ***Passenger Costs:*** passenger airport meals, passenger hotel costs and passenger compensations.

Finally, there is a less easily quantifiable cost that should also be included: the cost of delaying or canceling a flight from the passenger point of view. Most airlines use some kind of *rule-of-*

thumb when they are evaluating the impact of the decisions on passengers. Others just assign a monetary cost to each minute of delay and evaluate the solutions taking into consideration this value. We propose a different way of calculating this cost component as will be explained in Chapter 7 Section 7.5.2.

4.8 Brief Problem Statement

In a more informal way and without taking into consideration many details, we can say that, when faced with a Flight Delay (Equation 4.1) we want to:

- Eliminate or minimize the flight delay and the associated Costs, i.e.,

$$min \begin{cases} (Etd(f) - Std(f)) + Costs(f) & \text{, if } F_{dd} \\ (Eta(f) - Sta(f)) + Costs(f) & \text{, if } F_{ad}. \end{cases} \tag{4.2}$$

and we want to do that considering the following requirements:

- The three sub-parts of the problem, i.e., aircraft, crew and passenger should be considered at the same level of importance. We are looking for the best integrated solution.
- We should consider the local preferences and goals of each team on the AOCC. For example, the *Passenger Team* share some goals with the other teams (minimize the flight delay is one) but, at the same time, this team wants to reduce the Passenger Trip Time and the Costs related to the passenger part of the problem. At this time, we can say that the goal of each team is the following:

$$PassengerTeam \overset{min}{\rightarrow} (TripTime(f) + PaxCosts(f)) \tag{4.3}$$

$$CrewTeam \overset{min}{\rightarrow} (CrewDelay(f) + CrewCosts(f)) \tag{4.4}$$

$$AircraftTeam \overset{min}{\rightarrow} (Delay(f) + FlightCosts(f)) \tag{4.5}$$

- The solution should be found in real-time.
- We should take into consideration the dynamics of the existing information, i.e., the information can unexpectedly change.
- The information available may not be complete.
- The time available to get a solution might be a restriction.

As stated in the beginning of this section, this is a very brief, informal and not detailed problem statement. The complete problem and sub-problems will be formally defined in Chapter 7, together with our proposal for a *new approach for disruption management in AOCC*.

4.9 Chapter Summary

As we stated in Chapter 1 and considering the applied line of research we choose to follow, it is very important to describe well the real application domain scenario as well as the problem to be solved. This chapter is a step forward in achieving this goal. Here, we have presented the Airline Operations Control Problem (AOCP). To contextualize we gave some information about the preceding Airline Scheduling Problem (ASP) and, then, information about the AOCC organization

and information sources. The typical problems and events that cause the problems as well as the current disruption management process were detailed enough to allow a good understanding by the reader. We conclude the AOCP by showing the main costs involved as well as by providing a brief problem statement.

In the next chapter we will present *PORTO*, an improved Agent Oriented Software Engineering Methodology based on *GAIA* (Zambonelli *et al.*, 2003) that we have used to analyze and design the MAS we will propose in Chapter 7.

Part II
The Main Thing

Part II - The Main Thing it is the core of this thesis. It includes the research contributions of our work trying, at the same time, to prove the hypotheses presented in Section 1.3.2. It is structured as follows.

In **Chapter 5 - Porto - An Improvement to Gaia**, we propose an improvement to the *GAIA* methodology called *PORTO*, which is one of the main contributions of this thesis, and results from applying it to analyze, design and develop the MASDIMA - Multi-Agent System for Disruption Management (Chapter 7 and Appendix A). Each phase of the methodology is presented, using as an example the MASDIMA system. This will allow the reader to better understand the concepts together with the proposed methodology and, at the same time, get acquainted with the system we have developed as a proof of concept for our approach to the disruption management problem.

In **Chapter 6 - Generic Q-Negotiation Protocol**, we provide a formal definition of a negotiation protocol with adaptive characteristics, which is one of the main contributions of this thesis and will be used in the MASDIMA system as a decision mechanism. We also present two application examples in different application domains to show that the protocol is generic enough to be used in such heterogeneous environments. This chapter is our contribution to the *second hypothesis*.

In **Chapter 7 - A New Approach for Disruption Management in AOCC**, we present our new approach for disruption management in the airline domain, including how we represent the AOCC using a Multi-Agent System (MAS), a computational organization of intelligent agents. Besides providing new or updated processes and mechanisms for the AOCC we also aim to focus on the AOCC as an entity, trying to change the way AOCCs are setup and organized and not only in providing DSS tools. Our main focus in this chapter is on the conceptual aspects of our approach. This chapter is our contribution to the *first*, *third* and *fourth* hypotheses.

Finally, in **Chapter 8 - Experiments**, we present the experiments we have performed regarding the hypotheses presented in Section 1.3.2. We start by describing the experimentation scenarios under analysis as well as the metrics used. The approaches or methods we have used to perform the experiments are also presented as well as the results. The chapter ends with a discussion regarding the results obtained from the experiments.

Chapter 5
Porto - An Improvement to Gaia

Abstract This chapter[1] is the first out of the four main chapters of this dissertation containing original contributions. Here we propose an improvement to the *GAIA* methodology called *PORTO*, which is one of the main contributions of this thesis, and is the result of its application to the analysis, design and development of the MASDIMA - Multi-Agent System for Disruption Management. We present each phase of the methodology using as an example the MASDIMA system. This will allow the reader to better understand the concepts together with the proposed methodology and, at the same time, get acquainted with the system we have developed as a proof of concept for our approach to the disruption management problem.

5.1 Introduction

In Section 3.6, we have provided some background information about Agent Oriented Software Engineering (AOSE), including a short survey of the state of the art in this field. In this chapter we will present *PORTO*, a methodology that complements another, already proposed, methodology called *GAIA* (Zambonelli *et al.*, 2003).

It is important to have a methodology for the development of a software system, specially for systems that are complex (but not only). A software development methodology allows to structure, plan, and control the process of developing such a software system, resulting in systems that behave better and according to the requirements. A study conducted by NIST[2] in 2002 (Newman, 2002) reports that software defects cost the U.S. economy $59.5 billion annually. More than a third of this cost could be avoided if better software testing was performed. Additionally, it is commonly accepted that the earlier a defect is found the cheaper it is to fix it. Table 5.1 shows the cost of fixing the defect depending on the stage it was found according to (McConnell, 2004). For example, if a problem in the requirements is found only post-release, then it would cost 10 to 100 times more to fix than if it had already been found in the requirements review.

Table 5.1 Cost to fix a defect (McConnell, 2004)

		Time Detected				
		Requirements	Architecture	Construction	Testing	Post-Release
Time Introduced	Requirements	1x	3x	5 to 10x	10x	10 to 100x
	Architecture	–	1x	10x	15x	25 to 100x
	Construction	–	–	1x	10x	10 to 25x

Due to the above reasons and personal work experience of the author, we decided to use an AOSE to develop the MASDIMA system (see Chapter 7 and Appendix A) that supports the concepts and approach we propose in this dissertation. As such, this chapter proposes an improvement

[1] A previous version of this work was presented in *Antonio J.M. Castro and Eugenio Oliveira The Rationale Behind the Development of an Airline Operations Control Center using Gaia based Methodology* (Castro & Oliveira, 2008).

[2] National Institute of Standards and Technology

to the *GAIA* methodology, called *PORTO*, that capitalizes on the experience we gained during the analysis and design of MASDIMA. Particularly, this chapter points out:

- The rationale behind the analysis, design and implementation of our software system.
- How we have used a goal-oriented early requirements analysis to complement *GAIA*. We also present the advantages that we believe emerge with the use of our approach to the modeling phase.
- How we used the early requirements analysis to sub-divide the system into sub-organizations and how we have described and represented the environment model.
- How we created the preliminary role model as well as the interaction model and how we found useful to have a graphical representation of the preliminary role and interaction models together with the environment model, i.e., a *UML*[3] *combined diagram*.
- How we used the previous models to define the organizational structure of our system and how we represented it in UML, including how we mapped the abstractions to UML metaclasses plus the stereotypes we created.
- The steps we took to complete the role and the interaction models and how we represented those two models in UML including the mappings we done.
- How we defined the agent model and the service model and how we represented those models in UML including how we mapped the abstractions to UML metaclasses.
- How we included an implementation phase, with steps that allow to identify the concepts and actions as well as to perform the mapping between the services and the required behaviours.
- How we included a test and validation phase that allows to test and validate the system according to the requirements.

It is important to point out that the main goal of our work was not about *AOSE*. However, the contributions in this area appear due to the need we had to model a complex and realistic MAS.

The rest of this chapter is organized as follows. In Section 5.2 we present a quick overview of the *PORTO* methodology. In Section 5.3 we explain the *Requirements Analysis* phase and in Section 5.4 the *Analysis* phase, the two phases that allow to understand the system-to-be.

In Section 5.5 and 5.6 we present the *Architectural Design* and *Design* phases, respectively, that are related to the design of the system according to the requirements and specifications. In Section 5.7 we present processes and artifacts of the *Implementation* phase and in Section 5.8 the *Test and Validation* phase. We end with a chapter summary in Section 5.9.

5.2 Methodology Overview

As stated in the previous section the goal of *PORTO* is to complement the *GAIA* methodology with the inclusion of a *Requirements Analysis*, *Implementation* and *Test and Validation* phases in the process of building software, as well as the inclusion of new or replacement documents and artifacts. In Figure 5.1 we present an overview of the proposed methodology. The left column shows the phase name, the center column the processes that should be performed in each phase, and the right column, the documents and artifacts produced during each phase. A brief explanation of each phase of the proposed methodology, follows:

[3] Unified Modeling Language (http://www.uml.org)

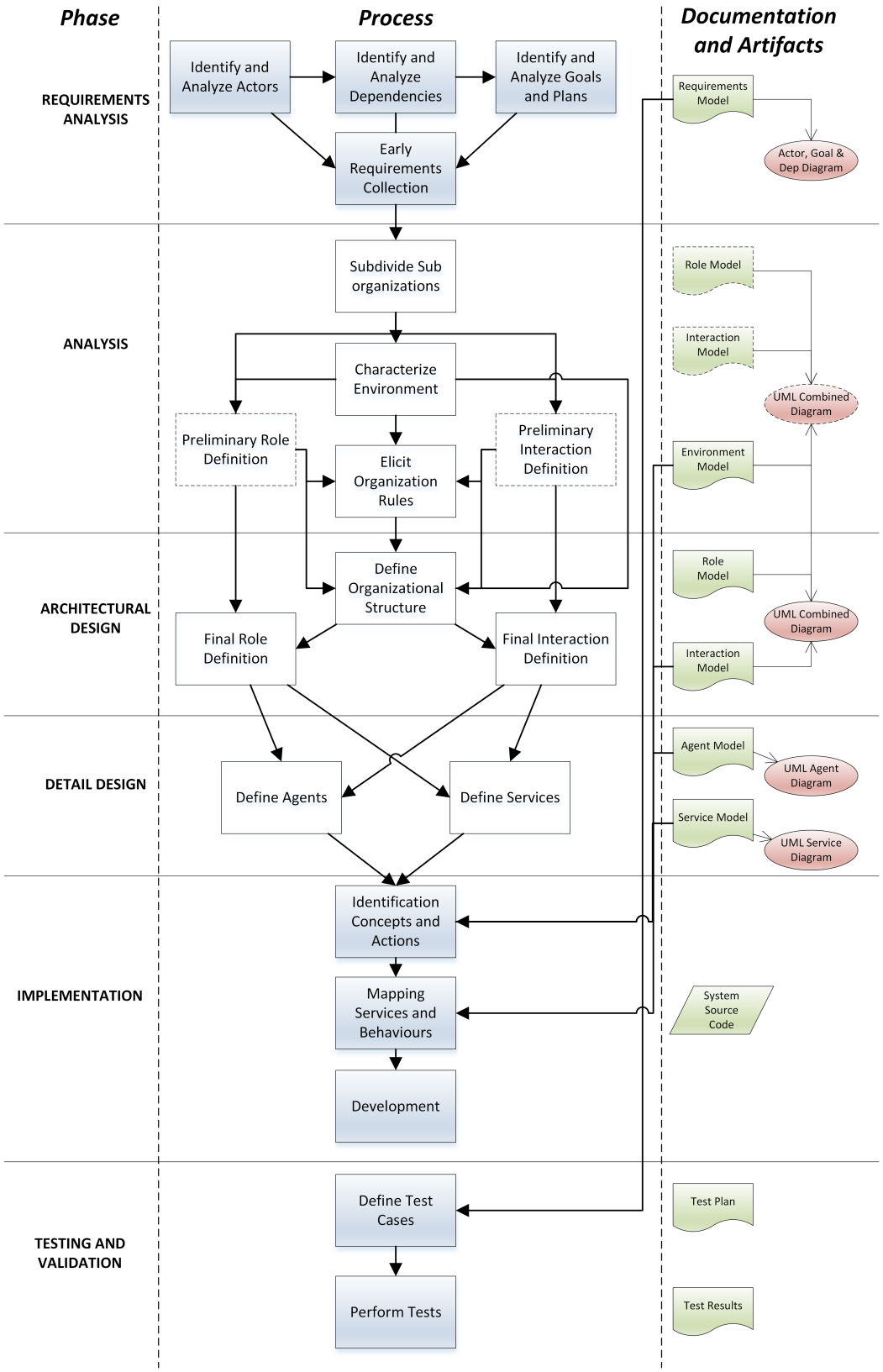


Fig. 5.1 Overview of PORTO Methodology

- *Requirements Analysis*: The goal of this phase is to understand the requirements from the point of view of the stakeholders[4] and users. *GAIA* does not propose a method to model the requirements and, as such, in *PORTO*, we propose to use a goal-oriented requirements analysis similar to the one presented in *TROPOS* (Bresciani *et al.*, 2004) and make the necessary changes to integrate it in the proposed methodology. The artifacts of this phase are the requirements model and the actor, goal and dependency diagrams.
- *Analysis*: The goal of this phase is to understand what the software system (here a MAS) will have to be, considering the requirements model. Like in *GAIA*, it will be produced an *environment*, *preliminary role* and *preliminary interaction* model as well as a set of *organizational rules*. This will be done considering the (possible) sub-organizations that might exist according to the requirements of the system-to-be. In *PORTO* we propose to use UML Diagrams as a notation to describe the several models produced in this phase (instead of the original ones) and introduce a new diagram called *UML Combined Diagram*.
- *Architectural Design*: The two previous phases are about understanding the MAS-to-be. Here, we start to make decisions about the actual characteristics of the MAS. This phase is not only about refining the roles and interaction models. It is also here that important decisions about the MAS organizational structure are made. As with the previous phase we follow the guidelines of *GAIA* regarding this phase, and we refine the *UML Combined Diagram* introduced in our approach.
- *Detail Design*: This phase is responsible for identifying the agents and services that will implement the roles, functions and interactions identified so far. It will take into consideration the spatial and physical distribution that is going to be adopted by the MAS. The outputs are the *agent* and *service* models. The specifications described in these models are neutral regarding the programming language or middleware used for implementation. We have replaced the agent and service model notation used by *GAIA*, by the *UML Agent* and *UML Services* diagram, respectively.
- *Implementation*: *GAIA* does not provide an implementation phase. In *PORTO* we propose to include some processes that will help the developers to identify the concepts and actions that need to be defined during implementation as well as a mapping of the services and agent behaviours. Although the steps presented here are valid for any programming language, we have used JAVA[5] and JADE[6] as an example to show how they can be applied. The output of this phase is the *system source code*.
- *Test and Validation*: During and after the implementation it is necessary to perform tests to see if the system works according to the specifications. This is a new phase that does not exist in *GAIA*. Here, the test cases are defined according to the specifications and the execution of these test cases will validate (or not) the system, showing if the system is working according to the requirements model. The *unit tests* are not considered at this stage but only in the development step of the implementation phase.

In the next sections we will detail each of the phases using as an example the analysis and design we have performed for the MASDIMA system (see Chapter 7 and Appendix A).

[4] A person, group or organization that has interest or concern in the system-to-be.

[5] http://www.java.com

[6] http://jade.tilab.com

5.3 Requirements Analysis

The *GAIA* methodology uses as an input a collection of requirements. The methodology does not propose a method to model these requirements. Although we could just list the requirements of the system-to-be, from interviews of the stakeholders and users, we believe it is important to have a better understanding of those requirements early in the process.

In *PORTO* we chose to adopt part of the goal-oriented requirements analysis of *TROPOS* (Bresciani *et al.*, 2004), i.e., the *Early Requirements Analysis*. In a goal-oriented requirements analysis the domain stakeholders are modeled as actors, depending on one another to achieve their goals. Plans to be performed and resources to be furnished are also modeled here. The key concepts or abstractions used in this phase are (Bresciani *et al.*, 2004):

1. *Actor*: Represents the stakeholders, the users of the system as well as the roles. It can be a physical, social or software entity that has strategic goals or concerns within the system or organizational setting.
2. *Goal*: Represents the actors' interests. There are *hardgoals* and *softgoals*. The former should be *satisfied* by the system, i.e., the system should have functionalities that allow the goal to be satisfied. The latter are non-functional requirements, i.e., the system does not necessarily implements functionalities to achieve the softgoals but might be operated in an environment (hardware and network infrastructure, for example) that will satisfy these softgoals. From now on and to simplify the reading, we will use the word goal as a synonym of hardgoal.
3. *Plan*: Represents a *way of doing something*. By executing a plan an hard or softgoal can be satisfied, i.e., a plan is a mean for satisfying a goal.
4. *Resource*: Represents an informational (e.g., a database) or physical (e.g., a sensor) entity.
5. *Dependency*: Dependencies exist between actors, meaning that one actor depends on another to achieve some goal, execute some plan or deliver/access some resource.

The *Requirements Analysis* phase has three main processes (see Figure 5.1), that will allow to collect all requirements in a *Requirements Model*. A description of these processes follows:

1. *Identify and Analyze Actors*: The objective is to identify and perform an analysis of all application domain stakeholders and users and their intentions as actors which want to achieve goals.
2. *Identify and Analyze Dependencies*: The objective is to focus on the dependencies between actors, modeling those dependencies as goals, plans or resources that depend on one or another to be achieved.
3. *Identify and Analyze Goals and Plans*: The idea is, from the point-of-view of the actor, perform an analysis of the actor's goals and plans, by using three basic reasoning techniques: *means-end analysis*, *contribution analysis* and *AND/OR decomposition*. The first one, aims at identifying plans, resources and softgoals that provide means to achieve a hardgoal. The second one, identifies goals that can contribute to the fulfillment of the analyzed goal. The last one, combines AND and OR decompositions of a root goal or plan into sub-goals or sub-plans, respectively, allowing to get a finer goal and plan structure.

By performing the above processes it is possible to summarize them in the *Early Requirements Collection* process.

Typically, to perform the above processes, the *requirements analyst* will interview all the potential stakeholders and users and read all the documentation that is available, to fully understand all the requirements. It is an iterative process, going back and forth as many times as needed.

The main graphical tool available for the analyst to help him/her is the *Actor, Goal and Dependencies Diagram*. Figure 5.2 shows a partial actor, goal and dependency diagram for the MAS-

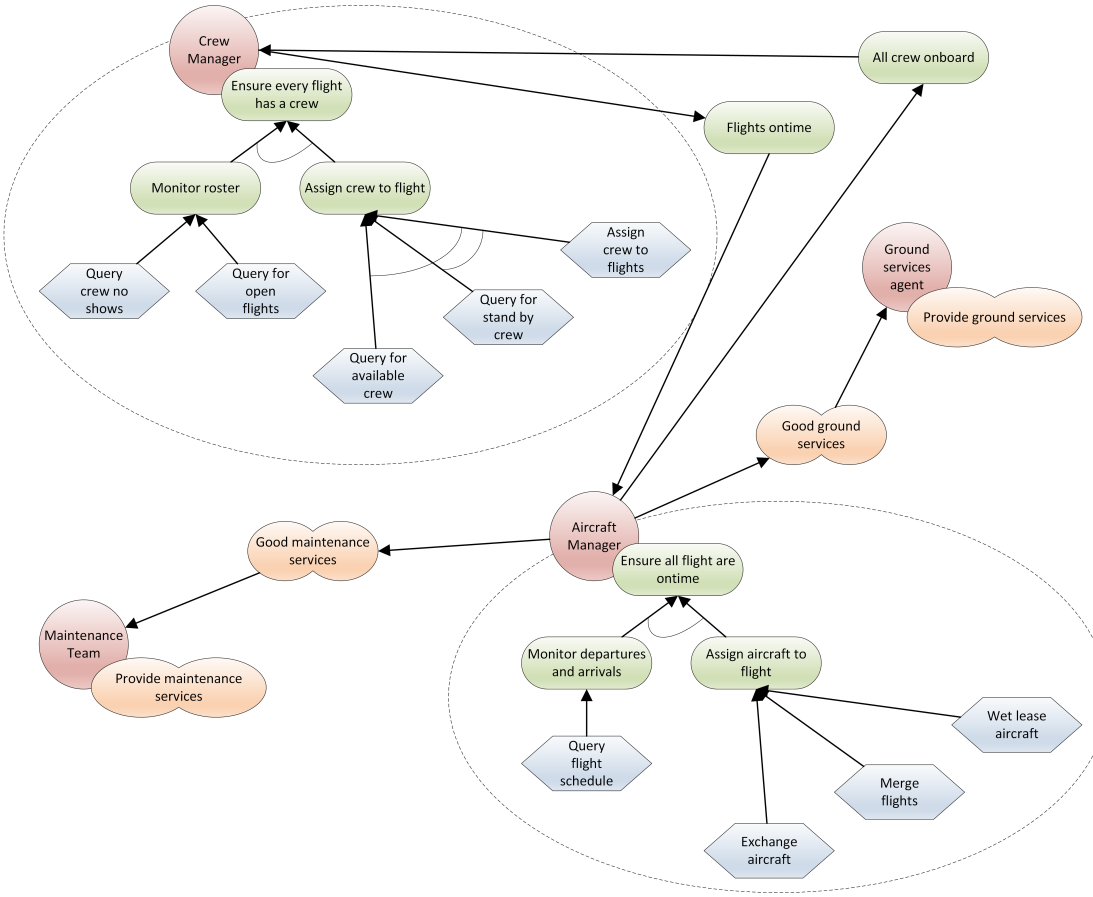


Fig. 5.2 Partial Actor, Goal and Dependency diagram

DIMA system, generated during the goal-oriented analysis. Here, the *red circles* represent actors, the *green ellipses* represent hardgoals, the *cloud ellipses* represent softgoals, the *blue hexagons* represent plans and the *large dash circle* represents the actor's perspective. The *and-decomposition* appears with a semi-circle connecting two arrows. Likewise, two arrows without a semi-circle mean a *or-decomposition*.

Looking at Figure 5.2, the main goal of the *Crew Manager* actor is to *Ensure every flight has crew*, meaning that before departure, it is necessary to guarantee that all flights have all crew members assigned according to the regulations. To be able to achieve this goal it is necessary to do an *and-decomposition*, meaning that it is necessary to achieve the sub-goals *Monitor roster* **AND** *Assign crew to flight*. To fulfill the sub-goal *Monitor roster* **any** of the following plans can be executed:

- *Query crew no shows*, meaning that it is necessary to query one of the resources available in the environment to obtain the name of the crew members that did not report for duty.
- *Query for open flights*, i.e., to query for flights with open crew positions.

To fulfill the sub-goal *Assign crew to flight* it is necessary to execute **all** of these three plans:

- *Query for available crew*, obtain a list of crew members that are available to be assigned to a specific flight. In this case, available means that the crew member does not have any kind of activity assigned, including a day off, and that, according to the regulations, can be assigned to the flight.
- *Query for stand by crew*, obtain a list of crew members that have assigned a stand by activity and that, according to the regulations, can be assigned to the flight.
- *Assign crew to flights*, finally, from the two lists obtained from the previous plans, to choose the best crew member according to criteria defined by the company, and assign him/her to the flight.

The execution of the referred plans is a mean to achieve the previously mentioned goals. To achieve its main goal the *Crew Manager* actor also depends on actor *Aircraft Manager*, through the dependency *Flights on time*.

Figure 5.2 also shows an example of a soft-goal, that is, a goal without a clear definition and/or criteria for deciding if it is satisfied or not (typically used to model non-functional requirements). Actor *Aircraft Manager* depends on actor *Maintenance Team* through the soft-dependency *Good maintenance services*.

5.3.1 Advantages

We found that the use of a goal-oriented analysis for eliciting the early requirements helped, not only in gathering and understanding the collection of requirements, but also in the analysis phase, namely:

- The modeling of the requirements, in terms of actors, their roles and their goals and dependencies among them, is more similar to the organization we have found in the airline AOCC, allowing us to better specify the software system-to-be.
- In subdividing the system: modeling the desires, intentions and dependencies of the stakeholders and specifying the system-to-be in terms of goals and softgoals, helped identifying the specific organizations and sub-organizations dedicated to the achievement of a given sub-goal. The teams in the AOCC exhibiting behaviours specifically oriented to the achievement of a goal and the corresponding mapping to sub-organizations are good examples of this advantage.
- In the preliminary role model: identifying the basic skills (functionalities and competences) required by the organization to achieve its goals. Again, the modeling of the system-to-be in terms of actors involved and their goals helped to identify the preliminary roles (basic skills). This is in accordance with the statement of the section 4.1.3 in (Zambonelli *et al.*, 2003).
- In the environment model: having a deeper understanding of the environment where the software must operate as well as of the interactions between software and human agents, during the modeling of the system-to-be, helped in identifying the roles of active components, allowing the distinction between active components that are only resources and others that should be agentified.
- In the preliminary interaction model: identifying the basic interactions that are required for the exploitation of the basic skills. This is not a direct advantage of applying a goal-oriented early requirements analysis. It appears in the identification of the basic skills (in the preliminary role model). However, having a better understanding of the actors and their dependencies, as

the result of the goal-oriented analysis, during the modeling of the system-to-be, facilitates the identification of the basic interactions.

5.4 Analysis

The objective of this phase is to understand what the system will have to be, considering the requirements model. From Figure 5.1 we see that it has five processes: (i) subdivision of the system into sub-organizations, (ii) characterization of the environment model, (iii) definition of the preliminary role model, (iv) definition of the preliminary interaction model and, (v) elicitation of the organizational rules. The outputs of this phase are the *preliminary role and interaction model* and the *environment model*. The main diagram used is the *UML Combined Diagram*.

Before proceeding to the explanation of each of the processes, it is important to define some of the abstractions used in this phase (and on the following ones) as well as the mappings we have done to the UML abstractions. This allows to better understand the processes and the UML diagrams used. The main abstractions and mappings are as follows:

- *Plans*: As stated in Section 5.3 it represents a *way of doing something*. We mapped this abstraction to a *UML class*. A class represents a group of things that have a common state and behavior. A class can represent a tangible and concrete concept, such as an invoice, or it may be abstract, such as a document. Using the class *attributes*, *methods* and *comments* abstraction we can represent the plan concept in a satisfactory way.
- *Resource*: Also as stated in Section 5.3 it represents an informational or physical entity. We found useful to map this abstraction to a *UML Table entity*.
- *Role*: We also mapped this abstraction to a *class*. A role represents functionalities and competences that need to be characterized. We found, at this point, that this mapping helped to visualize the roles and the relations among them. The role's safety properties are mapped as attributes and the activities as methods. However, as stated in (Bauer & Odell, 2005) "roles cannot be modeled in the necessary detail with any UML 2.0 diagrams". As we will show in Section 5.4.3 we use a role schema description table for each role to complete the information provided by the diagram. Nevertheless, we think that with our UML representation we are able to include almost all the necessary information.
- *Actions*: This concept represents the actions that roles or agents can perform on the environment resources (typically by executing a plan). For example, by reading or changing an informational resource. We mapped this concept to the *Dependency relationship* in UML adding the type of action to it.
- *Protocol*: The activities that involve interactions with other roles (protocols) are represented by an *Association relationship* in UML. We have created a protocol stereotype associated to the *association* metaclass in UML. Although some of these mappings might not be the appropriate one for the implementation phase, it did help us to visualize and model the organization with their roles, activities and protocols, using a widely used notation supported by several commercial tools.

5.4.1 Subdivide the system into sub-organizations

This is the first process to be performed in the Analysis phase. The objective is to see if the overall system can be subdivided into sub-organizations. Sub-organizations can be found when there are portions of the overall system that have **any of these conditions**:

- Exhibit a behavior specifically oriented towards the achievement of a given sub-goal.
- Interact loosely with other portions of the system.
- Require competences that are not needed in other parts of the system.

Continuing with MASDIMA as an example, we see that from the requirements analysis (see Figure 5.2) it is possible to determine three candidate sub-organizations that fulfill at least one of these conditions. The sub-organizations identified are described in table 5.2.

Table 5.2 Identification of sub-organizations

Sub-organization	Description
Crew Manager	The subgoal to achieve is *Ensure every flight has a crew*. It will loosely interact with other portions of the system because of the dependency *Flights on time* with the Aircraft Manager actor.
Aircraft Manager	The subgoal to achieve is *Ensure all flights are on time*. It will loosely interact with other portions of the system because of the dependencies *All crew onboard* with the Crew Manager actor and *Good maintenance services* with the Maintenance Team actor and *Good ground services* with the Ground Services handling agent actor.
Passenger Manager	The subgoal to achieve is *Ensure all passengers arrive at the destination*. Its interactions depend on *Flights on time* with the Aircraft Manager actor and *Good ground services* with the Ground Services handling agent actor. Please note that the Passenger Manager actor and dependencies are not included in the partial actor, goal and dependencies diagram on Figure 5.2.

5.4.2 Characterize the Environment

The second process of the analysis phase is to define the environment and, in this process, the first decision to be made is to distinguish between resources and active components. Resources might be, according to *GAIA*, ”variables or tuples, made available to the agents for sensing (e.g. reading their values), for effecting (e.g. changing their values) or for consuming (e.g. extracting them from the environment)”. On the other hand, active components are those components and services capable of performing complex operations with which agents in the MAS will have to interact. Computer-based systems or humans in a process are examples of active components that should not be treated as part of the environment but, instead, they should be agentified. In the MASDIMA case, we have one ”human” in the Operations Control Center that has an important role in the process (see Table 5.3) and to which the agents in the MAS will have to interact. This Active Component will be agentified through one agent. Table 5.4 shows some of the resources that are available in the environment.

Table 5.3 Active Components (partial)

Name	Description
Operational Control Supervisor	Final human authority regarding: initiating, canceling, consolidating or advancing flights, wet lease of airplanes, exchange of airplane, delay of flights, etc. In summary, this human has to authorize the application of any solution found in the operational plan.

Table 5.4 Resources (partial)

Name	Description
Crew Sign On	Contains information regarding the crew sign on for flights. It will be possible to know if a crew member did not report for duty. It will allow to implement the plan *query crew no shows* indicated in the crew manager goal diagram of Figure 5.2.
Pairings	Contains information regarding the pairings (and flights) that need to have crew members assigned. It will allow implementing the plan *query for open flights* and *assign crew to flights* indicated in the same diagram as the previous one.
Roster	Contains information regarding the roster of all crew members. It will allow implementing the plans *query for available crew*, *query for stand by crew* and *assign crew to flights* indicated in the same diagram.

For the environment model to be complete, we need to list or represent the resources and actions that will be performed to access them, from the environmental perspective. We proposed a UML representation of the environment model that includes the resources as well as the plans that will be executed using those resources. In our opinion, the inclusion of the plans allows to better model the environment.

Figure 5.3 shows a partial environment model taken from the MASDIMA. Regarding the *CrewSignON* resource we can see that the action will be to *read* information from attributes *dutyID* and *crewNumber* and the plan to be used is *QueryCrewNoShow*. That plan executes the *read* action performing the condition described in the comments, that is, returns the records where the *current date* is equal or greater than the *dutyDateTime* plus an additional time (in minutes) and where the *signOnDateTime* attribute is null (meaning that the crew did not report for duty). An example of an action that *changes* the resources is presented in the *Pairing* resource. It is possible to see that the plan *AssignCrewFlights* will *change* the attributes *cmdOpen*, *foOpen*, *csOpen*, *cfaOpen*, *faOpen*, subtracting to the existing value the number of crew members assigned. That same plan has *read* access to the *Roster* resource.

In the Preliminary Role Definition process we will start to use a UML Combined diagram that includes the environment model and the roles, showing the actions that those roles perform on the resources (through the execution of plans).

5.4.3 Preliminary Role Definition

In this process the objective is to identify the basic skills, that is, functionalities and competences, required by the organization to achieve its goals. Those basic skills are the preliminary roles and we need to identify the ones that will be played whatever the organization structure that will be

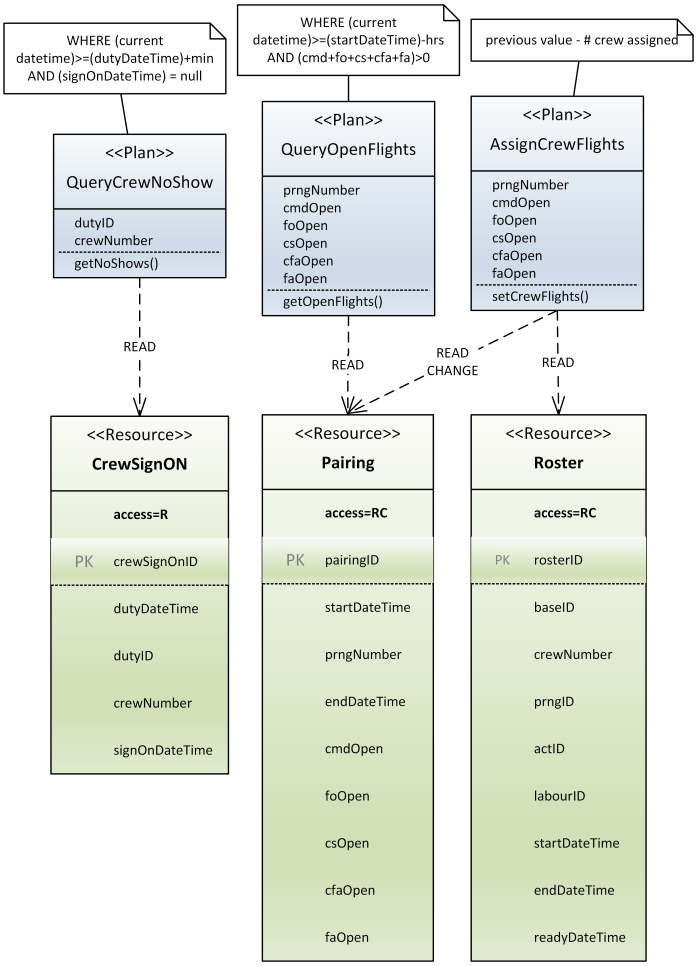


Fig. 5.3 Partial UML Environment Model Diagram

adopted later on during the Architectural Design phase. *GAIA* adopts an abstract, semiformal description to express the capabilities and expected behaviours of the preliminary roles. These are represented by two main classes: *Permissions* (actions allowed on the environment to accomplish the role) and *Responsibilities* (attributes that determine the expected behavior of a role, divided in Liveness Properties and Safety Properties). According to (Zambonelli *et al.*, 2003), *Liveness properties* "describe those states of affairs than an agent must bring about, given certain conditions". In contrast, *Safety properties* are *invariants*, i.e., an "acceptable state of affairs is maintained". The authors of *GAIA* propose the use of regular expressions to define these properties. For example, a *liveness expression* has the general form:

$$RoleName = expression$$

where *expression* are *activities* or *protocols*. Table 5.5 shows the operators used for *liveness* expressions. We recommend the reading of Section 4.1.3 of the *GAIA* methodology, for more information regarding these properties.

From the Actors, Goals and Dependencies diagram in figure 5.2, we can identify several roles that will exist independently of the final organization of our MAS. A partial list of those roles is:

Table 5.5 Operators for Liveness Expressions

Operator	Interpretation
$x.y$	x followed by y
$x\|y$	x or y occurs
$x*$	x occurs 0 or more times
$x+$	x occurs 1 or more times
x^w	x occurs indefinitely often
$[x]$	x is optional
$x\|\|y$	x and y interleaved

- *RosterCrewMonitor*, role associated with monitoring the crew roster for events related to crew members that do not report for duty and/or flights with open positions.
- *CrewFind*, role associated with finding the best crew member to be assigned to a flight after an event triggered by the *RosterCrewMonitor* role.
- *CrewAssign*, role associated with assigning the crew member found by the *CrewFind* role.

For each role it is necessary to define the permissions and the responsibilities and, an important step, it is necessary to identify any inconsistencies between what operations the environment allows and what the roles (agents) need or must be allowed to do. *GAIA* refers the need to create a *Role Schema* for each role, where these properties will be indicated. However, we found that a diagram that includes the Environment and Preliminary Roles will help to better identify these inconsistencies.

In the *UML Combined Diagram with Preliminary Roles and Environment* in figure 5.4 the *actions* allowed by the environment are labeled with an *R* for *reading* and a *C* for *changes*, for each resource through the *access* attribute. The *actions* the roles need or must be allowed to do are indicated by the *dashed arrows*. We can see that the *CrewAssign* role needs to read and change information from the resources *Pairings* and *Roster*. Looking to those resources representation, it is possible to see that these operations are allowed (both have the *access* attribute equal to *RC*). After analyzing this diagram we can see that there are no inconsistencies between the operations allowed by the environment and what the agents need to do.

To complete the preliminary role model, we have filled a role schema for each of the roles identified, with the information collected so far. It is possible to see an example in table 5.6.

It is important to point out that the *plans* that came from the requirements model and later represented in the environment diagram (see Figure 5.3), are replaced by *activities* performed by the roles. For example, the plans *Query crew no shows* and *Query for open flights* that allow the *Crew Manager* actor to reach goal *Monitor Roster*, are included in the activity CheckNewCrewEvents that appear in the *Role Schema*. In the UML combined diagram in Figure 5.4 these activities appear inside the role as methods with stereotype $<< act >>$.

The *permissions* in the role schema are represented as actions in the diagram and the *safety* rules as attributes inside the role. For example, attribute *conSignON* allows to validate rule $successful\_connection\_with\_CrewSignON$. The only thing that the UML Combined diagram cannot capture very well is the *liveness* expressions. As it is possible to see in Figure 5.4 we have used a comment to represent the liveness expression for the *RosterCrewMonitor* role.

The preliminary role model will be finished in the Architectural Design, giving its place to the full Role Model.

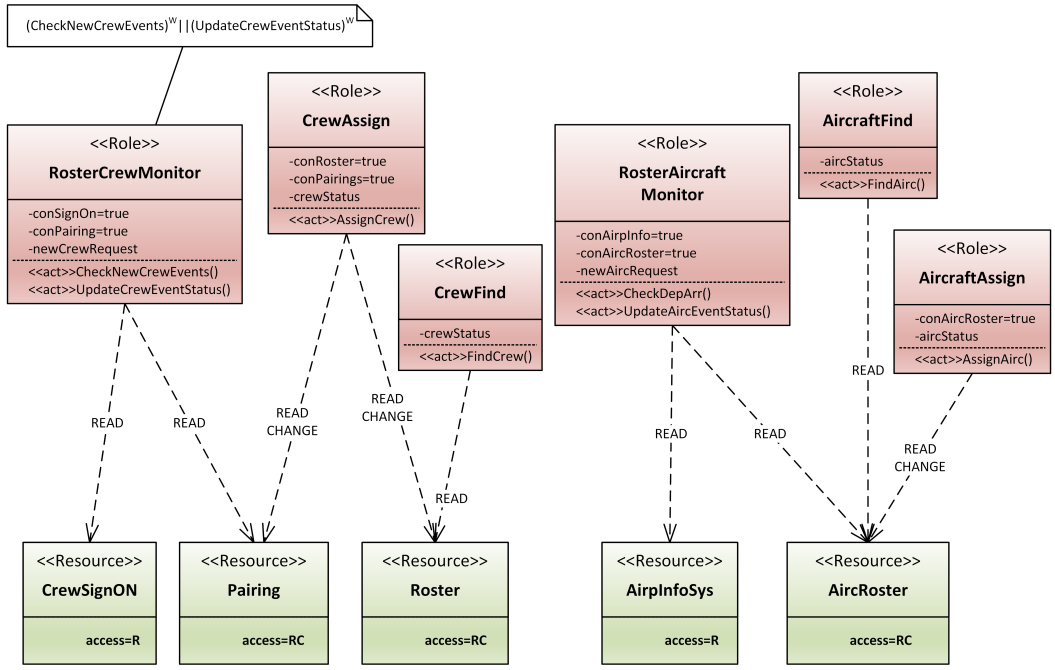


Fig. 5.4 UML Combined Diagram with Preliminary Roles and Environment (partial)

Table 5.6 RosterCrewMonitor preliminary role

Role Schema: RosterCrewMonitor

Description: This preliminary role involves monitoring the crew roster for events related to the crew members not reporting for duty and/or flights with open positions. After detecting one of these events it will elicit a solution from the organizer. It should be able to trace previous requests, avoiding duplicates, until it receives a message regarding the status of the request.

Protocols and Activities: CheckNewCrewEvents.UpdateCrewEventStatus

Permissions:
reads CrewSignON (to obtain all who did not report for duty)
reads Pairings (to obtain all flights with open positions)
Responsibilities:
Liveness:
$RosterCrewMonitor = (\underline{CheckNewCrewEvents})^W \| (\underline{UpdateCrewEventStatus})^W$
Safety:
$successful\_connection\_with\_CrewSignON = true$
$successful\_connection_w ith\_Pairings = true$
$new\_crew\_request <> existing\_unclosed\_crew\_request$

5.4.4 Preliminary Interaction Definition

The objective of this process is to capture the dependencies and relationships between the various roles in the multi-agent system organization. This is done with one protocol definition for each type of inter role interaction. *GAIA* proposes a very simple notation to specify the protocols, i.e., a table with the protocol name, description, initiator and partner role(s) and input and outputs of the protocol. We propose to use a UML Interaction diagram, since it is more expressive than the tabular notation.

Figure 5.5 shows the preliminary definition of the *requestCrew* protocol identified in the MAS-DIMA system. The protocol has the *RosterCrewMonitor* role as *initiator* and the *CrewFind* role as *partner*. After detecting an event, i.e., a crew member that does not report for duty or a flight with open positions (represented in the diagram by the parameter *crewEvents* in the *request* performative), the *RosterCrewMonitor* requests a solution to the *CrewFind* role (represented by the parameter *crewMembers* in the $inform-result$ performative). It starts by issuing a request and, then, the *CrewFind* has two alternatives: refuse to present a solution or accept the request. Accepting the request, if it does not have success in looking for a solution, it answers back with a *failure* performative that includes the reason. Having success, it informs when the job is done through the $inform-done$ performative and, then, sends the solution for the request through the $inform-result$ performative. This protocol is according to the *FIPA Request* protocol definition (FIPA, 2002c).

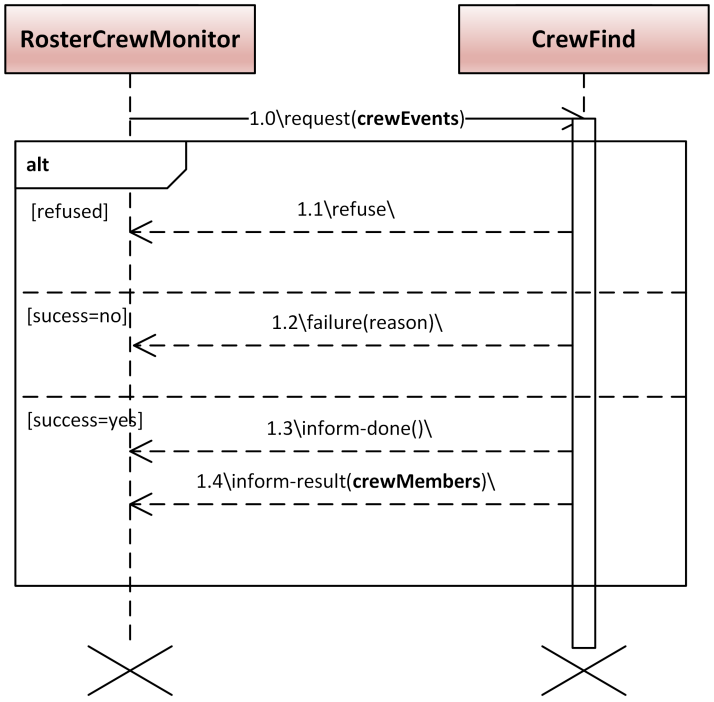


Fig. 5.5 UML Interaction Diagram for *requestCrew*

To have a better overview of the whole system, we found useful to complement the UML Combined diagram with the preliminary interactions (protocols) as presented in figure 5.6.

5.4.5 Elicit Organizational rules

According to the authors of *GAIA* "(...) there may be general relationships between roles, between protocols, and between roles and protocols that are best captured by organizational rules". Organizational rules are seen as responsibilities of the organization as a whole. In the preliminary role model we have already defined or approached the roles' responsibilities. As in that model, organizational rules also have safety and liveness rules, or, as in (Zambonelli *et al.*, 2003), constraints and relations, respectively:

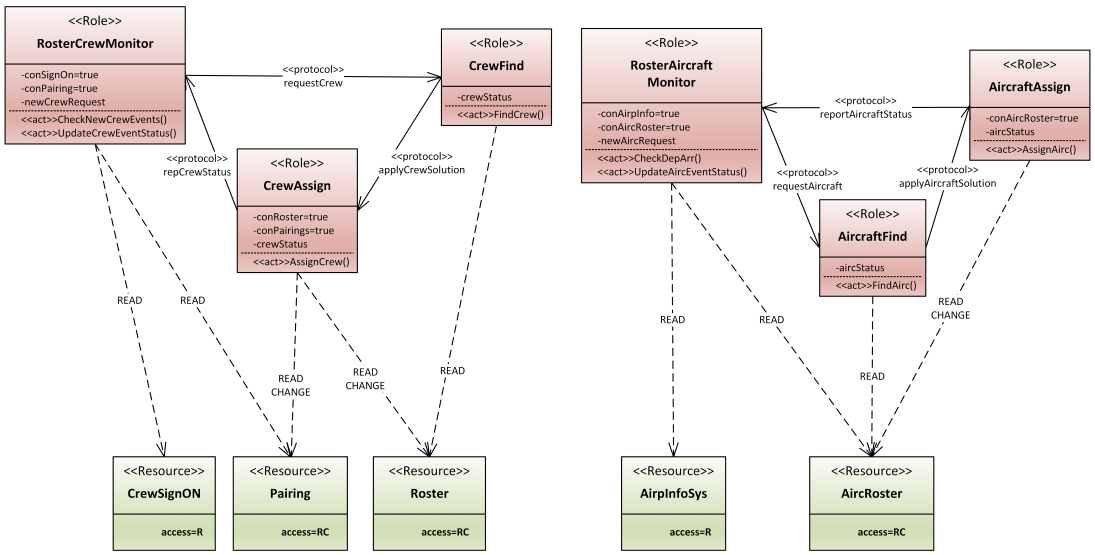


Fig. 5.6 UML Combined Diagram with Preliminary Roles, Protocols and Environment (partial)

- *Liveness organizational rules (relations)*, define "how the dynamics of the organization should evolve over time". For example, a specific role can be played by an entity only after it has played a given previous role.
- *Safety organizational rules (constraints)*, define "time-independent global invariants for the organization that must be respected". For example, two roles cannot be played by the same entity.

The formalism to express these rules can be the same used for the liveness and safety rules for the roles. They will be expressed by liveness and safety expressions respectively.

In summary, liveness expressions detail properties related to the dynamics of the organization, that is, how the execution must evolve and safety expressions detail properties that must always be true during the whole life of the MAS. A partial list of the liveness organizational rules (relations) we have defined for MASDIMA can be found in table 5.7 and in table 5.8 a partial list of the safety organizational rules (constraints).

Table 5.7 Liveness rules (relations)

Liveness rule or relations
$applyCrewSolution(CrewAssign(crew(x))) \rightarrow repCrewStatus(CrewAssign(crew(x)))$
Protocol *applyCrewSolution* must necessarily be executed by role *CrewAssign* for a specific crew solution *crew(x)* before role *CrewAssign* can execute protocol *repCrewStatus* for that crew solution.
$requestCrew(CrewFind(request(x))) \rightarrow applyCrewSolution(CrewFind(crew(x)))$
Protocol *requestCrew* must necessarily be executed by the role *CrewFind* for a specific request *request(x)* before role *CrewFind* can execute protocol *applyCrewSolution* for the solution found.

Table 5.8 Safety rules (constraints)

Safety rules or constraints	Description
$\neg(RosterCrewMonitor)\|CrewFind)$	Role *RosterCrewMonitor* and role *CrewFind* can never be played concurrently by the same entity.
$\neg(RosterCrewMonitor)\|CrewAssign)$	Role *RosterCrewMonitor* and role *CrewAssign* can never be played concurrently by the same entity.

5.5 Architectural Design

The analysis phase, presented on the previous section, has the objective of understanding what the MAS *will have to be*, i.e., what it is expected to do. The deliverables of the analysis (i.e., environment model and preliminary roles and interaction model) express the functionality and operational environment of the MAS. In the architectural design phase it is necessary to make decisions regarding the actual characteristics of the MAS. So, besides completing and refining the preliminary models, the design will rely in actual decisions about the organizational structure and in modeling the MAS based on the specifications produced.

The architectural design phase has three processes: (i) define the organizational structure; (ii) define the final role model and (iii) define the final interaction model. The outputs of this phase are the role and interaction model and the main diagram used is the UML Combined diagram.

The choice of the organizational structure is very important and will affect the development of the succeeding phases. For that we need to choose the desired topology and control regime to be applied. The *GAIA* paper (Zambonelli *et al.*, 2003) has a very good explanation of this important step. Another very useful reading is Mark S. Fox's paper on organizational theory (Fox, 1981).

5.5.1 Defining the organizational structure

To define the organizational structure we will continue to use as an example the MASDIMA system. From the specification documents of the analysis phase, we found the following main requirements to consider for defining the organizational structure:

- *From the Actors, Goals and Dependencies diagram*: The main organization (AOCC) has three sub-organizations, that is,
  1. Aircraft Manager.
  2. Crew Manager.
  3. Passenger Manager.
- *From the Environment model*: We have identified the following active component (resources that will be agentified), that is, Operational Control Supervisor (human authority).
- *From the Preliminary Role model*: We have identified a requirement that the *CrewFind* role and *AircraftFind* role use different techniques to find the solutions.
- *From the Organizational rules*: We have identified the roles that cannot be played concurrently by the same entity, such as (this is a partial list), *RosterCrewMonitor* and *CrewFind*, *RosterCrewMonitor* and *CrewAssign*, *AircraftFind* and *PaxFind*.

Having this information we defined the organization structure of our MAS. Table 5.9 gives a summary of the topologies and control regimes applied.

Table 5.9 Topologies and control regime

Organization	Topology	Control Regime
AOCC	Multilevel hierarchy	Mixed: cooperative and authoritative.
Crew Manager	Multilevel hierarchy	Work specialization.
Aircraft Manager	Multilevel hierarchy	Work specialization.
Passenger Manager	Hierarchy	Work specialization.

The control regime was defined following the guidelines of (Zambonelli *et al.*, 2003; Fox, 1981). In the AOCC organization we have cooperative control regime between the *Managers'* roles due to the *peer relation* among them, and authoritative from *Operational Control Supervisor* to the *Managers'* roles due to the *control relationship* (for example, *Operational Control Supervisor* controls *Aircraft Assign* role). The work specialization control regime on the other organisations is derived from the fact that the role (for example, *CrewFind* or *AircraftFind*) provides specific services.

To represent the organizational structure, *GAIA* suggests a coupled adoption of a formal notation and of a more intuitive graphical representation. Table 5.10 is a formal notation of the organization structure that we defined for the *Passenger Manager* sub-organization. Please note that the relationship types identified here are neither mutually exclusive (for example, a control relation type may also imply a dependency relation type), nor complete (other types of relations may be identified). A (partial) UML representation of the organization structure is presented in figure 5.7 following the suggestions of (Bauer & Odell, 2005).

Table 5.10 Organization structure for passenger recovery

Statement/Comment
$\forall i, OperationalControlSupervisor \overset{control}{\rightarrow} PaxApply[i]$
This means that the role *OperationalControlSupervisor* has an authoritative relationship with role *PaxApply*, controlling, in this case, all the actions of role *PaxApply*. Specifically, role *PaxApply* needs approval from *OperationalControlSupervisor* before applying the solution. Please note that role *OperationalControlSupervisor* is shared between this sub-organization and *Aircraft Manager* sub-organization.
$\forall i, OperationalControlSupervisor \overset{depends\_on}{\rightarrow} PaxFind[i]$
This means that the role *OperationalControlSupervisor* relies on resources or knowledge (a solution found to solve a passenger problem) from the role *PaxFind* to accomplish its task (i.e., to authorize or not authorize the assignment of a specific solution).
$\forall i, j, PaxFind[i] \overset{depends\_on}{\rightarrow} PaxMonitor[j]$
This means that the role *PaxFind* relies on resources or knowledge (an event related to a passenger problem) from the role *PaxMonitor* to accomplish its task (i.e., to find a solution to the passenger problem).

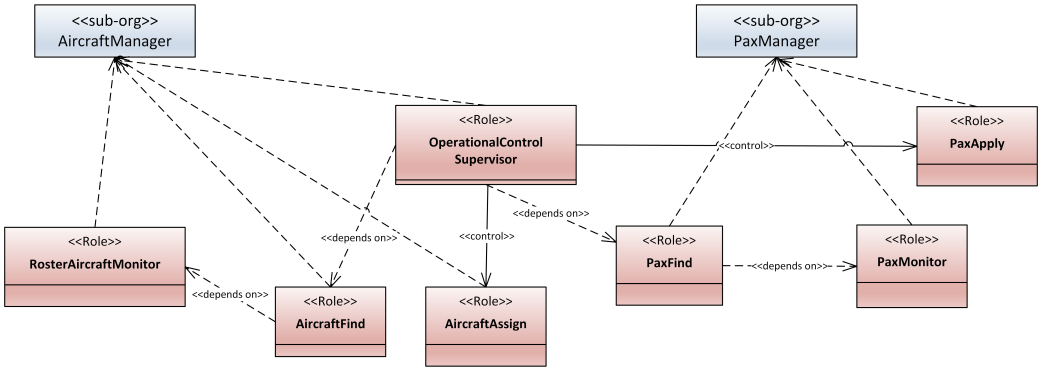


Fig. 5.7 Organization Structure (partial) in a UML Diagram

To be able to represent the organization structure in UML we made some mappings between the abstractions used here and the UML artifacts as well as created some stereotypes, as follows:

- *Organization Abstraction*: We have mapped this abstraction to a package. In UML packages provide a way to group related elements. Using package diagrams it is possible to visualize dependencies between parts of the system. If we see an organization as a package we can take advantage of these characteristics and model them using package diagrams. For the goal of the organization we can use a note or constraint to represent it. We have created a sub-organization stereotype associated to the package metaclass of UML.
- *Depends on Abstraction*: We have mapped this abstraction to a dependency relationship. The dependency relationship in UML is the weakest it is possible to define. A dependency between classes means that one class uses, or has knowledge of, another class. They are typically read as "... uses a ...". A *depends on* relation between two roles usually means that one role relies on resources or knowledge from the other role. We have created a *depends on* stereotype associated to the dependency metaclass of UML.
- *Controls Abstraction*: We have mapped this abstraction to an association relationship. Association relationships in UML are stronger than dependencies and typically indicate that one class retains a relationship to another class over an extended period of time. They are typically read as "... has a ...". A control relation between two roles usually means that one role has an authoritative relationship with the other role, controlling its actions. We have created a control stereotype associated to the association metaclass of UML.
- *Peer Abstraction*: We have also mapped this abstraction to a dependency relationship. A peer relation between two roles usually means that they are at the same level and collaborate to solve problems. We have created a peer stereotype associated to the dependency metaclass of UML.

5.5.2 Completing the role and interaction model

After having the organization structure it is possible to complete the role and the interaction model. Some roles' interactions result from the organization topology and the protocols that need to be executed from the control regime defined. The tasks that are necessary to be performed to complete both models are:

- Complete the *activities* in which a role is involved, including its liveness and safety responsibilities.
- Define *organizational roles*, that is, those whose presence was not identified during analysis and that result directly from the adopted organization structure.
- Complete the definition of protocols specifying which roles the protocol will involve.
- Define *organizational protocols*, that is, those whose identification derives from the adopted organization structure.

It is important to notice the distinction between characteristics that are *intrinsic* from the *extrinsic*. *Intrinsic* are independent of the use of the role and/or protocol in a specific organization structure. *Extrinsic* are the ones that derive from the adoption of a specific organizational structure. This distinction is important in terms of reuse and design for change.

Table 5.11 RosterCrewMonitor role (Final)

Role Schema: RosterCrewMonitor

Description: Monitors the crew roster for events related to crew members not reporting for duty and/or flights with open positions. After detecting one of these events, it will request a solution from the organization. Traces previous requests and avoids duplicates, until it receives a message regarding the status of the request.

Protocols and Activities: CheckNewCrewEvents, UpdateCrewEventStatus, requestCrew, repCrewStatus

Permissions:
reads CrewSignON (to obtain all who did not report for duty)
reads Pairings (to obtain all flights with open positions)
changes CrewEvents (keep log of events)
Responsibilities:
Liveness:
$RosterCrewMonitor = (\underline{CheckNewCrewEvents}^W.requestCrew)^W \,||\, (repCrewStatus^W.\underline{UpdateCrewEventStatus})^W$
Safety:
$successful\_connection\_with\_CrewSignON = true$
$successful\_connection_with\_Pairings = true$
$successful\_connection_with_CrewEvents = true$
$new\_crew\_request <> existing\_unclosed\_crew\_request$

Table 5.11 is an example of a final role specification taken from the MASDIMA complete role model. It is important to point out the differences from the final role schema when comparing with the preliminary one (from table 5.6). First, the liveness responsibilities were completely defined and, second, due to the work done during the architectural design, a new resource was identified: *CrewEvents*. This new resource will keep a record of events status. The permissions property was updated to reflect the access to this new resource. The liveness property specifies what activities and protocols the role will have. They express part of the role's expected behavior. In our example, *activities* appear underlined and express actions performed by the role that do not involve interaction with any other role (similar to a method in object oriented terms). *Protocols* are activities that do require interaction with other roles. From the liveness expression of our example

$$RosterCrewMonitor = (\underline{CheckNewCrewEvents}^{W}.requestCrew)^{W} \parallel (repCrewStatus^{W}.\underline{UpdateCrewEventStatus})^{W}$$

We can see that role *RosterCrewMonitor* consists of executing the activity CheckNewCrewEvents indefinitely (marked by the *W* operator), followed by the execution of the protocol *requestCrew*. Both of these are performed indefinitely. In parallel (marked by the $\parallel$ operator) it executes the protocol *repCrewStatus* indefinitely followed by the activity UpdateCrewEventStatus. Both also performed indefinitely.

Regarding the final interaction model, Figure 5.8 shows an example taken from the MASDIMA complete interaction model. The important thing to point out in here is the distinction that is made

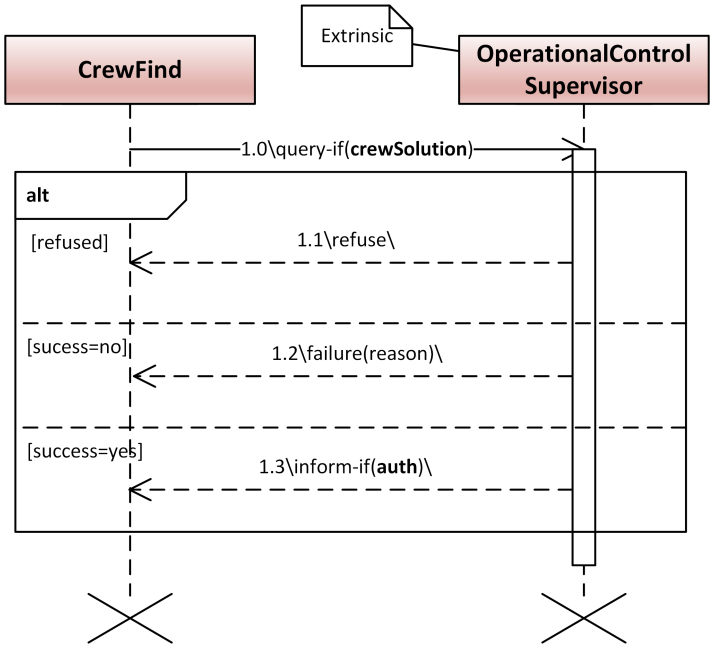


Fig. 5.8 UML Interaction Diagram for *sendCrewSolution*

regarding the extrinsic characteristic. In this specific example, it is the *OperationalControlSupervisor* partner that is specific to the organization structure defined. This information might be important if we, later, decide to change the organization structure.

At this stage it is desirable to draw a UML Interaction Diagram similar to the one in figure 5.8, for all protocol definitions of our interaction model. Finally, the preliminary UML Combined Diagram from the Analysis phase, should also be updated with the final role and interaction model. A partial example of such a diagram taken from MASDIMA is presented in Figure 5.9.

5.6 Detailed Design

This phase is responsible for identifying the agents and services that will implement the roles, functions and interactions identified so far. It will take into consideration the spatial and physical distribution that is going to be adopted by the MAS.

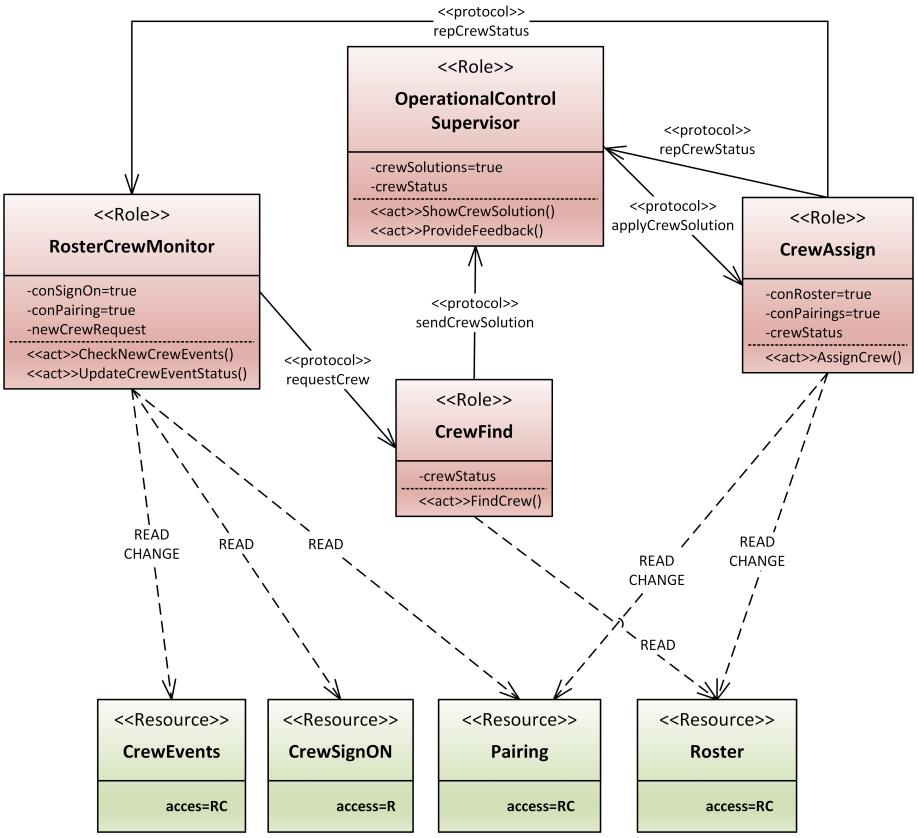


Fig. 5.9 UML Combined Diagram (partial) with final roles, interaction and environment

The Design phase has two processes, i.e., define agents and define services and the outputs are the Agent model and the Service model. They will show the agents that will be implemented as well as the services that will be necessary to implement by each one of them. It will be a programming language/middleware neutral specification. This phase uses two UML diagrams: the UML Agent Diagram and the UML Services Diagram.

5.6.1 Define Agents

To build the agent model it is possible to make a one-to-one correspondence between roles and agent classes. However, there are some advantages in trying to find a better mapping. The better one is to try to compact the design by reducing the number of classes and instances leading to a reduction in conceptual complexity. This has to be done without: affecting the organizational efficiency, violating the organizational rules and creating ”bounded rationality” problems (that is, without exceeding the amount of information it is possible to process in a given time). *GAIA* does not specify any special notation for showing the agent model although it implicitly suggests the adoption of a class model diagram. A simple way of representing the Agent Model is to use a table like the one we present in 5.12.

Table 5.12 Agent Model (partial)

Agent classes/roles
$OpMonitor^{1..n} \overset{play}{\rightarrow} RosterCrewMonitor, RosterAircraftMonitor, PaxMonitor$ This means that agent class *OpMonitor* will be defined to play the roles *RosterCrewMonitor*, *RosterAircraftMonitor* and *PaxMonitor*, and that we will have between one and *n* instances of this class in our MAS (*n* depends on the functional and physical distribution adopted).
$OpAssign^{1..n} \overset{play}{\rightarrow} CrewAssign, AircraftAsssign, PaxApply$ This means that agent class *OpAssign* will be defined to play the roles *CrewAssign*, *AircraftAssign* and *PaxApply*, and that we will have between one and *n* instances of this class in our MAS (*n* depends on the functional and physical distribution adopted).
$OpSupervisor^{1} \overset{play}{\rightarrow} OperationalControlSupervisor$ This means that agent class *OpSupervisor* will be defined to play the role *OperationalControlSupervisor*, and that we will one instance of this class in our MAS.

5.6.2 Define Services

The services derive from the protocols, activities and liveness expressions of the roles that each agent implements. Usually, there will be one service for each parallel activity of execution that the agent has to execute. According to *GAIA*, the service model requires that, for each service that may be performed by an agent, four properties are identified: inputs, outputs, pre-conditions and post-conditions. The *inputs* and *outputs* are derived from the interaction model and from the environment model. If the service involves elaboration of data and the exchange of knowledge between the agents, they will come from the protocols. If the service involves evaluation and modification of the environment resources, they will come from the environment. The *pre* and *post conditions* represent restrictions on the execution and completion, respectively, of the services. They derive from the role safety properties as well as from organizational rules. Applying the above guidelines we obtain the service model. In table 5.13 we represent some services for agent class *OpMonitor* and in table 5.14 some services for agent class *OpSupervisor*, both from the MASDIMA example.

5.6.3 UML Representation

As we stated before, *GAIA* does not propose any notation to represent the Agent and Service model besides the tabular simple notation we presented in the previous section. In this section we are going to show how we have represented this two models using a UML diagram. Figure 5.10 shows the UML Agent Model (partial and simplified) we have defined for the MASDIMA and Figure 5.11 shows the UML Service Model (partial and simplified) for the agent class *OpMonitor*.

To do that we have adopted the following mappings between the methodology abstractions and UML concepts:

Table 5.13 Services (partial) agent class OpMonitor

Service Description
Service: Monitor Crew Events. *Input*: current date, crew slack time, pairing slack time. *Output*: A list of dutyID, crewNumber, prngNumber, listOpenPositions, eventID. *Pre-condition*: Successful connection with CrewSignON and Pairing resources. *Post-condition*: A new crew event that has to be different from an existing unclosed one.
Service: Update crew event status. *Input*: eventID, eventStatus. *Output*: Number of records updated. *Pre-condition*: Successful connection with CrewEvents resource. *Post-condition*: Successful update of the CrewEvents resource.

Table 5.14 Service (partial) agent class OpSupervisor

Service Description
Service: Obtain crew solution authorization. *Input*: List of crew members to be assigned. *Output*: Authorization status (OK or NOT OK). *Pre-condition*: At least one crew solution found. *Post-condition*: User confirms or does not confirm authorization.
Service: Request crew solution application. *Input*: Authorized list of crew members to be assigned. *Output*: Request status (YES = solution can be applied. NO = solution cannot be applied). *Pre-condition*: Authorization status = OK. *Post-condition*: User sees status of the request on the screen.

- *Agent Class Abstraction*: We have mapped this abstraction to a class and created an *Agent Class* stereotype associated to the class metaclass in UML. To identify the roles that each agent class implements we have created a *role* stereotype associated to the property metaclass in UML. The instances of the agent class are represented using *Constraints*.
- *Services Abstraction*: We mapped the services abstraction to an *interface*. In UML an *interface* is a classifier that has declarations of properties and methods but no implementations. It provides a contract that a classifier that provides an implementation of the interface must obey. The inputs are represented as properties of the interface and the method represents the service that needs to be implemented. The outputs are what the method returns. For example, the interface *MonitorCrewEvents* represents the *service* with the same name and attributes *currentDate*, *crewSlackTime* and *pairSlackTime* are the inputs. *CheckNewCrewEvents* is the operation to be implemented. The agent class *OpMonitor* realizes that interface by providing an implementation for the operations and properties (the dashed line starting at the agent class to the interface, with a closed arrowhead at the end, shows this realization).
- *Pre and Post-conditions*: We present the pre/post-condition using constraints, associated to the specific interface. For example, the *MonitorCrewEvents* interface has the following pre-condition and post-condition, respectively:

$$connCrewSignON = true;\ connPairings = true$$

$$new\_event <> existing\_event$$

It is important to note that *Invariant Constraints*, that is, constraints applied to all instances of the class are not reflected in this diagram. It is possible to do it by applying domains, attribute types, attribute multiplicity and valid values of attributes.

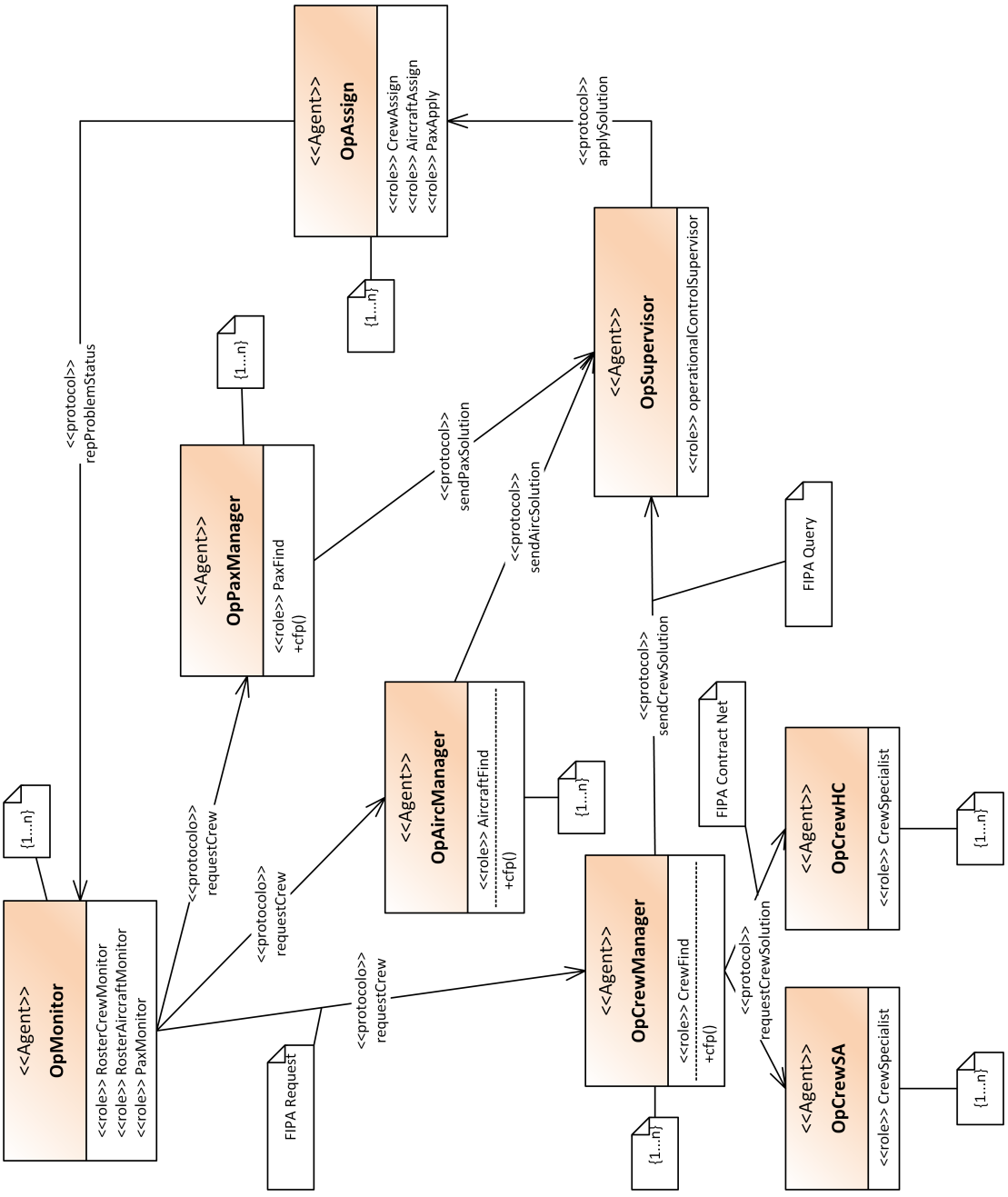


Fig. 5.10 Simplified UML Agent Model (partial)

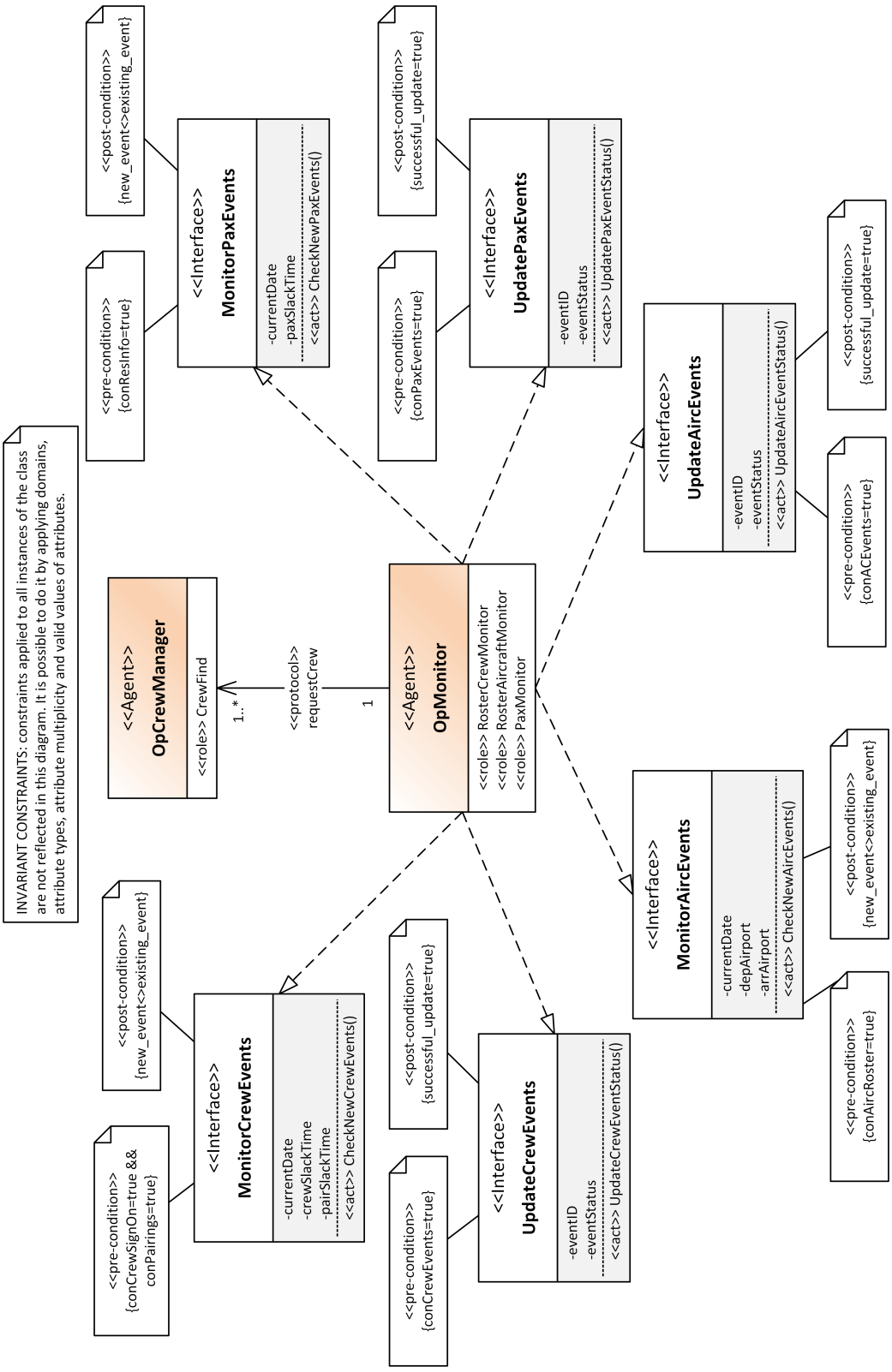


Fig. 5.11 Simplified UML Service Model (partial)

5.7 Implementation

GAIA methodology does not include tools for describing an Implementation phase. According to its authors "after the successful completion of the design process, developers are provided with a well defined set of agent classes to implement and instantiate, according to the defined agent and services model". This section is our approach to include an implementation phase in the methodology and is the result of the experience we had in developing the MASDIMA system.

The implementation phase includes three processes: (i) identification of concepts and actions; (ii) mapping of services to behaviours and (iii) development. The output of this phase will be the *system source code*. Although these processes are generic enough to be used independently of the programming language used, here we are going to use JAVA and JADE (Bellifemine *et al.*, 2004) as examples, because they were the language and framework we used to implement MASDIMA.

JADE is a software development framework written in Java language aimed at the development of MAS. JADE also works as a distributed agent platform across several hosts. Another important feature is that JADE is a FIPA[7] compliant agent platform and provides implementations of agent communication language (ACL) messages between agents as well as standard interaction protocols (such as FIPA-request, FIPA-query, etc.).

To start the implementation it is necessary to map between the detailed design obtained from *GAIA* and the language/middleware used. We have defined four tasks to be performed:

1. Model the interaction between the several agents (in terms of communications and how to represent the content of messages), identifying the proper concepts and actions and defining them as classes, deciding which of the methods (serialized objects or extensions of predefined classes) will be the ideal to use.
2. Define a notation to be used for the names of Agents, Services and Protocols according to the implementation language and their best practices.
3. Relate each one of the services in the service model to the possible behaviours to use, according to the necessary activities to be performed.
4. Define for each one of the interactions protocol in the model, the necessary performatives to use and why. It should also reflect the choice of using a standard protocol or the choice of building a new one.

The following sections explain how the above tasks were included in each of the processes defined.

5.7.1 Identification of Concepts and Actions

In this process the first task is to define the vocabulary and semantics for the content of the messages that will be exchanged by the agents in the system. This can be different according to the programming language and middleware used for implementation. In the case of JADE and Java, it provides three ways to implement communication between agents regarding the content of the messages:

1. The use of strings.
2. Transmission of serialized Java objects.
3. Ontology classes taking advantage of the standard FIPA format.

[7] http://www.fipa.org

Before deciding which of the methods to use and after reviewing the interaction, environment, agent and service models, we can identify the necessary concepts and actions. Some of the concepts and actions, taken from the MASDIMA, are represented in Table 5.15. In MASDIMA we chose to pass the content of messages as objects. However, if we are modeling an *open* MAS, where the agents need to interoperate with agents designed and implemented by different system designers, the best choice might be to use ontology classes.

Table 5.15 Some concepts and actions from MASDIMA

Concepts	**Description**
CrewEvent	Characterizes a crew event that initiates the process of finding a crew solution.
CrewSolutionList	Characterizes a list of crew solutions proposed by the agents that are specialists in crew problems to the *OpCrewManager* corresponding to the *CFP* initiated after a crew event has been detected.
CrewSolution	Characterizes the crew solution chosen by agent *OpCrewManager*, that will be presented to the *OpSupervisor* for authorization.
Actions	**Description**
ApplyCrewSolution	Action of applying the crew solution after it has been authorized.
UpdateEventStatus	Action of making the status update of a crew/aircraft or passenger event.

The second task to be performed in this process is to define the implementation notation to use for agents, protocols and services. During the design phase, the notation used is defined according to the modeling tool used (in the case of *PORTO* is UML) and the names used for the concepts reflect that choice. However, for the implementation, it is important to follow the programming language and middleware guidelines. So, in this task, besides defining the notation we also map between the names used in the design and the new names defined for implementation. Table 5.16 presents a partial list of the notations and mappings defined for the MASDIMA system.

At the end of this process we get the following:

1. A list of concepts and actions that, using the chosen programming language, need to be implemented.
2. An implementation notation for the concepts, agents, protocols and services according to the programming language and middleware best practices and guidelines.
3. A mapping between the names used in design time and the names chosen for implementation.

5.7.2 Mapping Services to Behaviours

The goal of this process is threefold:

1. Map the services that need to be implemented to the behaviours provided by the programming language and middleware that are more suitable to be used.
2. Choose between using a standard protocol and/or implement new ones.
3. Considering the choice made regarding the protocols, define the performatives to be used.

Table 5.16 Agents, Protocols and Services notations (partial) from MASDIMA

Design Name	Implementation Name
Agents	
OpMonitor	MonitorAgent
OpAircManager	AircraftManagerAgent
OpCrewManager	CrewManagerAgent
OpPaxManager	PaxManagerAgent
OpCrewSA	CrewSASpecialistAgent
OpCrewHC	CrewHCSpecialistAgent
OpAssign	SolutionAssignAgent
OpSupervisor	SupervisorAgent
Protocols	
requestSolution	request-solution
sendAircSolution	send-aircraft-solution
sendCrewSolution	send-crew-solution
sendPaxSolution	send-pax-solution
requestCrewSolution	crew-solution-negotiation
applySolution	request-apply-solution
Services	
MonitorCrewEvents	MonitorCrewEvents
UpdateCrewEvents	UpdateCrewEvents
RequestSolutionApplication	RequestApplySolution

Regarding the first and second, the choice of behaviours (or other similar concept) and standard protocols is limited by the programming language and middleware used. In the case of MASDIMA all services will be implemented with JADE behaviours that will *run* inside or extend a JADE *CyclicBehaviour*. This is necessary because all agents will be running indefinitely, as it is possible to infer from the liveness expressions of the roles that each agent represents. The agents will perform indefinitely some services (for example, monitoring) and/or waiting for a message to act (for example, messages that initiate interactions protocols that they need to be part of). Regarding the standard protocols and as stated before, JADE is FIPA compliant and, as such, the FIPA standard protocols are available. Table 5.17 shows a partial list of the mappings between services and JADE behaviours and FIPA protocols.

Table 5.17 Mapping (partial) of JADE behaviours and Services in MASDIMA

Service: MonitorCrewEvents *JADE Behavior:* Ticker *FIPA/JADE IP:* fipa-request *Protocol implementation name:* request-solution
Service: FindCrew *JADE Behavior:* Simpler *FIPA/JADE IP:* fipa-request; fipa-contract-net *Protocol implementation name:* request-solution; crew-solution-negotiation

Regarding the third, the performatives are related to the interaction protocols used. So, if the designer chooses to follow a standard (like FIPA-ACL performatives) there is no need to create

new performatives. As an example, in the MASDIMA system we used the FIPA-ACL performatives and in the case of the *crew-solution-negotiation* protocol we used the *FIPA contract net*. In this case, the performatives are: cfp, refuse, propose, $reject\_proposal$, $accept\_proposal$, failure, inform (done), and inform (result).

5.7.3 Development

The last process in the implementation phase is the development of the software system. Having all the information collected from the previous steps it is possible to use a suitable development environment to start coding. It is important to point out that it is in this phase, that the *unit tests* are performed and not in the Test and Validation phase. *Unit Testing* is a method by which individual units of source code are tested to determine if they work as expected. As such, they are part of the development phase. Developers should follow the best practices and guidelines of the development tool and programming language used to implement the system.

In Appendix B, Section B.1 we present some examples of how we have implemented in the MASDIMA system, some of the concepts defined previously, i.e:

- The *CrewEvent* concept.
- The agent *MonitorAgent* that implements the *RosterCrewMonitor*, *RosterAircraftMonitor* and *PaxMonitor* as well as the *Monitor* and *Update* services associated, implemented using JADE behaviours *Ticker* and *OneShot*.
- The *MonitorCrewEvents* through the JADE *TickerBehaviour*.

The output of this process is the System Source Code that, after compilation, can be tested and validated as we propose in the Test and Validation phase.

5.8 Test and Validation

The Test and Validation phase (or Verification and Validation) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. In some literature it may also be referred to as Software Quality Control. Although it seems similar, *verification* and *validation* are different concepts. The definition of these two concepts, according to the *Capability Maturity Model* (Paulk *et al.*, 1993), is as follows:

- *Verification*: The process of evaluating software to determine whether the products of a given development phase satisfy requirements imposed at the start of that phase.
- *Validation*: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

The goal of *verification* is to ensure that the system has been built according to the requirements and design specifications, i.e., to ensure that the *right thing was built*. *Validation* has the objective of ensuring that the system meets the needs of the users and stakeholders, and that the specifications were correct, i.e., to ensure that *it was built right*. *Validation* confirms that the system, as provided, will fulfill its intended use.

In this phase we have two processes: definition of the test cases and execution of the test cases. The outputs will be the test plan for the first process and the test results for the second one. This phase ends when the test results are positive for each and every test case.

5.8.1 Define Test Cases

Test cases are prepared for *verification*, i.e, to determine if the process that was followed to develop the final system is right. To fully test that all requirements of a system are met, it is necessary to define test cases that cover all requirements, testing not only the return defined according to the requirements but, also, return that is not defined on the requirements.

As one might expect, it is important to keep track of the link between the requirement and the tests. One way of doing that is to use a *traceability matrix* (Carlos, 2012). A traceability matrix is a document, usually in the form of a table, that relates each requirement to the test cases.

The written test cases should include the following (an example is presented in Table 5.18):

- A description of the functionality to be tested.
- The preparation required to ensure that the test can be conducted.
- Known input (tests a precondition).
- The test step number if applicable.
- The related requirement or requirements.
- The name of the person that performed the test.
- Expected Result, i.e., the result that should be obtained according to the requirements (tests a post-condition).
- Actual Result, i.e., the result after the test has been performed.
- Test result, i.e., pass or fail.
- Remarks about the execution of the test.

As we stated in the Implementation phase, the test cases designed here are not the *Unit Tests* designed and performed during implementation. The goal of the *Unit Tests* is to test specific parts of the code and the test cases aim at testing the response of the system according to the requirements specification.

The output of the *Define Test Cases* process is *Test Plan*, i.e., the set of all test cases designed to test the system. As one might expect, the most time consuming part is the creation of the tests and modifying them when the system changes. The *Test Plan* will be used by the system testers to validate the system.

5.8.2 Perform Tests

After getting the *Test Plan*, it is possible to perform the final process in this *Test and Validation* phase, i.e., validation. As we stated before, the objective is to validate if the system is built according to the requirements of the user.

The first thing to do, is to assign software testers to each test case, so that they can perform the tests. It is important to point out that someone on the team (typically the quality control or software test manager) should keep track of the tests, the person who performed it and the result.

Table 5.18 Test case example from MASDIMA

Test Case ID:	MON-023
Tester:	Pedro Rica
Summary Functionality:	Detect flight problem and ask request solution.
Requirement(s):	RQ-017 and RQ-021
Preparation:	Flights in operation should appear in *Flight Monitoring* window.
Step number:	01
Step Input:	An event should create a flight departure delay.
Step description:	Look at the *Flight Problem* window.
Step Expected Result:	The flight number, schedule departure time, expected delay, number of violations and unsolved status, should appear.
Step Actual Result:	The same as the expected result.
Step number:	02
Step Input:	An event should create a flight departure delay.
Step description:	Look at the *Flight Map* window.
Step Expected Result:	The flight affected should have a red circle blinking.
Step Actual Result:	The same as the expected result.
Step number:	03
Step Input:	The flight with the unsolved problem should appear in the *Flight Problem* window.
Step description:	a) Click in the flight number. b) On the *Solution* window click on the *Supervisor Default Values* tab.
Step Expected Result:	The correct supervisor default values should appear for each dimension.
Step Actual Result:	The same as the expected result.
Step number:	04
Step Input:	The flight with solved status should appear in the *Flight Problem* window.
Step description:	a) Click in the flight number. b) On the *Solution* window click on the *Solution Proposal* tab. c) On the *Solution* window click on the *Solution Plan* tab.
Step Expected Result:	a) Should appear values for delays and cost for each dimension as well as the solution utility. b) The actions to be applied in the operational plan should appear for the dimensions.
Step Actual Result:	The same as the expected result.
Test Result:	PASS
Remarks:	In step 02 the blinking should stay for, at least, 10 seconds.

The second thing to do is to perform each test case. The software tester should assign *PASS* or *FAIL* in the *Test Result* as well as any remarks on the *Remarks* field. Finally, having all the tests performed, the *software test manager* should review the test results with the rest of the team and pass the information so that the tests that failed can be corrected.

5.9 Chapter Summary

In this chapter we have presented an *AOSE* methodology called *PORTO*, that results from complementing the *GAIA* (Zambonelli *et al.*, 2003) methodology. As we stated in the introduction section, the main goal of our work was not about *AOSE*. However, the contributions in this area appear due to the need we had to model the MASDIMA system, a complex and realistic MAS.

When compared to *GAIA* our approach has the following main differences:

- *Requirements Analysis*: *GAIA* does not have a requirements analysis phase. We have adapted the early requirements analysis of the *TROPOS* (Bresciani *et al.*, 2004) methodology to be included here (Section 5.3). The advantages that emerged from including this phase are presented in Section 5.3.1.
- *Notation*: We have used *UML* to replace or complement the notation proposed by *GAIA's* authors. This has the advantage of replacing an unfamiliar notation by a standard and more familiar notation used by software developers. Specifically, we have replaced or complemented the following:
  - Replace the table notation for the protocol definition by UML Interaction Diagram as presented in Figure 5.8 (Section 5.5).
  - Replace the formal notation representing the organizational structure by a UML Class Diagram as presented in Figure 5.7 (Section 5.5.1).
  - Replace the table representation of the agent model by a UML Class Diagram as in Figure 5.10 (Section 5.6.1).
  - Replace the table representation of the service model by a UML Class Diagram as in Figure 5.11 (Section 5.6.2).
  - A new diagram, called *UML Combined Diagram* that includes the environment, interaction and role model as well as the organization structure (Section 5.4.3). This diagram has the advantage of helping to better visualize the organization with their roles, activities and protocols.
- *Implementation*: *GAIA* does not have an implementation phase. Using our experience in implementing the MASDIMA system we added this phase to the methodology (Section 5.7).
- *Test and Validation*: Like with the previous one, *GAIA* does not have this phase. Again, using the experience obtained in developing the MASDIMA system, we propose to include this phase in the methodology (Section 5.8).

In the next chapter we will present another of the main contributions of our work: *Generic Q-Negotiation Protocol*, a negotiation protocol with adaptive characteristics, that will be used later in our work.

Chapter 6
Generic Q-Negotiation Protocol

Abstract Chapter 5 was about the agent oriented software engineering methodology called PORTO which we used to model the multi-agent system that supported our proposal. In this chapter the main goal is to introduce formal definitions of a negotiation protocol with adaptive characteristics, called Generic Q-Negotiation (GQN), which is one of the main contributions of this thesis and will be used later on in our work. Additionally, we provide two application examples with different domains to show that the protocol is generic enough to be used in such heterogeneous environments. We end the chapter by drawing the attention to some advanced features that, although not defined here, we think should be included in the future. The work presented in this chapter will help us to draw conclusions about the *second hypothesis* as formulated in Section 1.3.2.

6.1 Introduction

According to Smith (Smith, 1980) Distributed Problem Solving (DPS) is ”the cooperative solution of problems by a decentralized and loosely coupled collection of knowledge sources (KS’s), located in a number of distinct processor nodes”. It is cooperative because the individual KS’s do not have sufficient information to solve the problem and it is decentralized because both control and data are geographically and/or logically distributed. Loosely coupled, in this context, means that individual KS’s spend most of their time on computation rather than communication. With the advent of the agent paradigm, Lesser et al. (Durfee *et al.*, 1989) defined Cooperative Distributed Problem Solving (CDPS) in a similar way, giving a special focus to problems or constraints that ”problem solver nodes” face when working together. The first problem or constraint is related to the way nodes must coordinate their asynchronous problem solving to build compatible solutions for their interdependent sub-problems. The second is related to the limitations on communication between nodes due to bandwidth constraints, meaning that the nodes must have local reasoning to decide on appropriate actions and interactions and be capable of changing its behavior according to the circumstances. Due to advances that happened in the last 20 years on communication and computer power, the latter of the constraints does not represent such a critical problem as in 1989. Nevertheless, it is still very important that the nodes (or agents in an agent and multi-agent paradigm) have the capability to adapt to the environment. In our opinion the former constraint still requires the researchers’ attention. We believe that our proposed protocol addresses these two constraints in an efficient way.

It is typical to make a distinction between multi-agent systems (MAS) and distributed systems (DS) (Wooldridge, 2009d). Agents in MAS may have been designed and implemented by different individuals and possess different goals and preferences, making them self-interested. Agents are also assumed to be acting autonomously, making decisions at runtime, instead of having the decisions defined at design time. This implies that the agents need to be capable of dynamically coordinate their activities and cooperate with others. In DS the agents share a common goal and

coordination and cooperation are typically defined at design time. Due to the assumption of cooperation in distributed problem solving (these systems are assumed to implicitly share the same goals); CDPS and DPS in general, have been considered to belong to DS research area (see for example (Wooldridge, 2009d)). On the other hand, MAS have focused more on the study of societies of self-interested agents and, as such, cannot be assumed to share the same goals.

Despite this earlier distinction and in our opinion, it is possible to use the MAS paradigm to model both types of environments as well as an environment whose agents have both competitive and cooperative characteristics (we named it mixed environment). Assuming an environment composed by one organizer (the agent that encompasses the problem and starts the resolution process) and several respondent agents (those which have the expertise to solve the problem or part of it) we might say that the environment is competitive when the organizer defines a goal and the respondent agents compete to achieve that goal and win. That is, the respondent agents are self-interested (have goals of their own) and perform their actions in order to arrive at the goal set by the organizer. Each respondent agent must, however, be able to arrive at the proposed goal exclusively on its own. Nevertheless, if the agents do effectively cooperate during the resolution process, and thus produce the organizer's goal result together (be it through the distribution of tasks and/or information, etc.), then it is appropriate to name this a cooperative environment.

In this chapter we present the GQN protocol (Generic Q-Negotiation) for multi-attribute negotiation with several rounds and qualitative feedback with interdependent dimensions, which is suitable for cooperative (including distributed), competitive and mixed environments. It supports several types of agents: those that are willing to cooperate (with or without some degree of self-interestedness and rationality), as well as those that are competitive (self-interested and rational). The GQN protocol also supports a MAS with a mixed environment: it is cooperative because the agents, individually, might not have the full expertise to achieve the goal or to solve the problem (completely); it is competitive, because the agents have their own interests and preferences and want to achieve their personal goals, optionally competing with agents with the same expertise. Finally, it supports functional distribution as well as spatial distribution, either logical (data partition, for example) or geographical and it is suitable for use on very dynamic environments.

This protocol supports agents that act as organizers and/or respondents, assuming more than one role (initiator and participant) with different problem solving methods and bid strategies. Additionally, agents are able to learn (adapt) their strategies during bid formulation, due to the inclusion of a Q-Learning algorithm (Watkins & Dayan, 1992). Our model is multidimensional because each agent represents and includes knowledge about one or more dimensions. Each dimension is characterized by a set of attributes representing a part of the problem. The set of dimensions represents and describes the problem and the set of partial solutions, one for each dimension, represents a solution to the problem.

At this moment, the GQN protocol has only been tested in closed environments; meaning that all the agents that form the MAS are known and designed fully *a priori*, as well as their interactions' protocol, imposed at design time. Nevertheless, our protocol supports some characteristics of the open environments. For example, it is possible to introduce in our MAS a new agent, designed by another designer, and with a different strategy. The only new information this agent needs to know about is the interaction protocol.

To start explaining our GQN Protocol we have to begin by an exam of the assumptions behind our model (Section 6.2), those that format our understanding of the main concepts involved and their definitions (Section 6.3). We will describe the external features (the environment on which

agents interact), which are, the Communication and Domain Language (Section 6.4), the Interaction Protocol (Section 6.5) and the Rules of Interaction (Section 6.6) as well as the internal features of the participating agents, which are, the Internal Motivations and the Decision Mechanisms of the Organizer and Respondent agents (Sections 6.7 and 6.8). Our chapter structure was inspired by the reading of the paper by (Rahwan *et al.*, 2004).

Additionally, a theoretical analysis of the protocol is provided (Section 6.9) and ee show how the protocol could be used to model two very distinct applications, one with a typical competitive environment and another with a more cooperative and distributed environment (Section 6.10. The chapter ends with the identification of some advanced features (Section 6.11) and a summary (Section 6.12).

6.2 Assumptions

Rationality: Agents are rational in the sense they will try to maximize their individual utilities and act according to their preferences. However, they are not purely rational in the sense given by the Rational Theory (Homans, 1973; Coleman, 1973) specifically regarding two of rational theory assumptions:

- An agent (individual) has full or perfect information about exactly what will occur as a consequence of any choice made, and,
- An agent (individual) has the cognitive ability and time to consider and weigh out every possible outcome of any decision made.

The rationality of the agents in our model may be limited by the finite amount of time they have to make their decisions and, possibly, by the limited information they have. Considering this, we may say they have Bounded-Rationality (Simon, 1955).

Benevolence: We do not assume that our agents are benevolent, i.e., there is no *a priori* disposition for the agents to be helpful. Our opinion, as well as the opinion of (Wooldridge & Jennings, 1999) is that cooperation does not imply benevolence. For an agent to be cooperative it does not mean that it is obliged to cooperate or to have such a priori disposition. Our agents are autonomous, they can choose to participate or not in cooperative activities. In our protocol, if cooperation exists, that is implicit, i.e., it is a consequence of the negotiation.

6.3 Definitions

6.3.1 Related to the Object of Negotiation

Definition 6.1. *Problem*
A Problem P is the object of negotiation and is represented as a n-tuple of the dimensions that describe subparts of the problem:

$$P = \langle d_1, \ldots, d_{n-1}, d_n \rangle \tag{6.1}$$

where n is the number of dimensions of the problem.

Definition 6.2. *Dimension*
A subpart of a problem represented by an *id*, a set of attributes A, a set of attributes domain I, an optional set of dependencies between dimensions RT, a set of preferred values for the attributes VP and a set of attributes score functions V:

$$d_i = \left\langle id^i, A^i, I^i, \left[RT^i\right], VP^i, V^i \right\rangle \; with \; d_i \in P, i \in \aleph . \tag{6.2}$$

The sets VP^i and V^i are private to the agent meaning that the other agents will not have access to them.

Definition 6.3. *Partial-Solution*
A partial-solution ps is a possible solution to a dimension of a problem. It is represented by the attribute-values av_i regarding the correspondent dimension d_i:

$$ps_i = \langle av_1, \ldots, av_{m-1}, av_m \rangle \; with \; ps_i \in S, i \in \aleph \tag{6.3}$$

where m is the number of attributes in ps_i.

Definition 6.4. *Solution*
A solution S represents a possible solution to a problem and is a n-tuple of the partial-solutions for each dimension of the problem, plus an optional solution utility value uv for the agent that presented the solution:

$$S = \langle ps_1, \ldots, ps_{n-1}, ps_n, [uv] \rangle \tag{6.4}$$

where n is the number of partial-solutions of the solution.

Definition 6.5. *Feedback*
A qualitative feedback given by the organizer agent regarding each attribute value of every partial-solution, during proposal evaluation and classification:

$$F = \left\langle \langle f_1, \ldots, f_{m_i} \rangle_1, \ldots, \langle f_1, \ldots, f_{m_i} \rangle_n \right\rangle \; with \; f_k \in QF \tag{6.5}$$

where m_i is the number of attributes of a dimension i and n the number of dimensions. The set QF can be defined by the system designer according to specific application needs. However, this set should have, at least, those members that allow the classification of an attribute as being as expected, lower or higher than expected. For example, $QF = \{ok, low, high\}$.

6.3.2 Related to the Negotiation Algorithm and Agent's Characteristics

Definition 6.6. *Negotiation Model*
A negotiation model with GQN can be seen as a n-tuple:

$$NegMod = \langle O, R, E, CL, DL, IP \rangle \; with \; IP = \langle SM, RI \rangle \tag{6.6}$$

O is the set of organizer agents (i.e. each one responsible for defining what a specific problem is and to divide it into subparts, as well as, to initiate the negotiation and to evaluate the proposals received), R is the set of respondent agents (i.e., responsible for presenting proposals to the organizer), E represents the environment (e.g., resources available), CL represents the communication

language (i.e., to facilitate the agent's interactions during negotiation.), *DL* the domain language (i.e., a language to convey concepts related to the environment, proposals, the different agents, time and the object of negotiation) and the interaction protocol *IP* (i.e., the sequence of messages exchanged by the agents during the negotiation process - *SM*, and the rules of interaction - *RI*).

Definition 6.7. *Organizer Agent Negotiation Process*
From an organizer agent point of view a negotiation process $NegP^o$ is represented as a n-tuple composed by the organizer agent o, a set of respondent agents R, a problem P and an ordered set L^o of negotiation messages (i.e., a log with the sequence of proposals received together with the proposals' evaluation sent):

$$NegP^o = \langle o, R, P, L^o \rangle \, with \, o \in O \tag{6.7}$$

Definition 6.8. *Respondent Agent Negotiation Process*
From a respondent agent point of view a negotiation process $NegP^r$ is represented as a n-tuple composed by the respondent agent r, an optional set of respondent agents Rb with which he has to negotiate to complete its proposals (if it does not possess competences in all dimensions of the problem), an organizer agent o, a problem P, an ordered set L^r of the negotiation log (i.e., the sequence of proposals sent as well as the comments received including the feedback) and a n-tuple Q with information related to the q-learning algorithm used as part of the agent strategy (see also Definitions 6.17,6.18):

$$NegP^r = \langle r, [Rb], o, P, L^r, Q \rangle \tag{6.8}$$

$$with \, o \in O, \; r \in R, Rb \subset R \; and \; Q = \langle state, action, qvalue \rangle$$

Definition 6.9. *Organizer Agent Negotiation Log*
A negotiation log of an organizer agent o is an ordered set L^o of tuples with the information of the received proposal (i.e., a possible solution S) as well as the evaluation ev assigned by the organizer agent to the proposal, in each round:

$$L^o = \left\{ \langle S_1, ev_1 \rangle, \ldots, \left\langle S_{p-1}, ev_{p-1} \right\rangle, \left\langle S_p, ev_p \right\rangle \right\} \; with \; p \in \aleph, ev \in \Re \tag{6.9}$$

p denotes the number of proposals received.

Definition 6.10. *Respondent Agent Negotiation Log*
A negotiation log of a respondent agent r is a set L^r of tuples with the information of the proposals sent (i.e., a possible solution S) as well as the comment c given by the organizer agent to the proposal in each round. It also includes the list RF of requests and informs/failures received during the inter-RA negotiation (i.e, the negotiation that happens between respondent agents during each round of the main negotiation):

$$L^r = \left\{ \langle S_1, c_1, RF_1 \rangle, \ldots, \left\langle S_{p-1}, c_{p-1}, RF_{p-1} \right\rangle, \left\langle S_p, c_p, RF_p \right\rangle \right\} \; with \; c_p \in \{winner, F\} \tag{6.10}$$

F is the feedback according to definition 6.5 and p denotes the number of proposals sent.

Definition 6.11. *Competence of an Agent*
For the agents to participate in a negotiation they must have specific Competence(s), i.e., the scope of its knowledge or ability. The competence C of an agent a is characterized by an id, a set of attributes CA, a set of attribute domains CD and a scoring function CV that evaluates the preferred solutions of agent a:

$$C_a = \langle id_a, CA_a, CD_a, CV_a \rangle \tag{6.11}$$

Please note that an agent can have more than one competence.

Definition 6.12. *Organizer Agent Utility Function*
Let o represent an organizer agent and r a respondent agent. Additionally, let V^p represent the value of the offer according to a score function. If $O^t_{r\to o}$ represents an offer from r to o at round t, we define the utility of agent o as:

$$U_o\left(O^t_{r\to o}\right) = 1 - V^p\left(O^t_{r\to o}\right) \tag{6.12}$$

Definition 6.13. *Round Winner*
In a negotiation where an organizer agent o negotiates with a set of respondent agents R concurrently, agent r_i's offer is better than agent r_j's offer if:

$$U_o\left(O^t_{r_i\to o}\right) > U_o\left(O^t_{r_j\to o}\right) \tag{6.13}$$

Agent r_i is the *winner of the round t* iff $\exists i, \forall j \neq i$ and condition 6.13 is true.

Definition 6.14. *Coherent Solution*
A coherent solution S is one where every partial-solution ps has been issued by a competent agent a and, if there are restrictions rt, such partial-solution complies with those restrictions, i.e.:

$$Coherent\,(S) \Leftrightarrow S = \langle ps_1, ps_2, \ldots, ps_n\rangle \wedge \tag{6.14}$$

$$\forall i \in [1..n]\,(Proposed\,(a, ps_i) \wedge Competent\,(a, d_i) \wedge$$

$$\forall j \in [1..k_i]\,(\neg Restriction\,(rt_j, d_i) \vee (Restriction\,(rt_j, d_i) \wedge Complies\,(ps_i, rt_j))))$$

Definition 6.15. *Restriction*
A restriction ϕ over a partial-solution is a conjunction of constraints imposed on the attribute values of that partial-solution. A constraint is a binary relation of the form $\alpha\theta\gamma$, where $\theta \in \{>, <, \leq, \geq, =\}$, $\alpha \in A$ (set of attribute names) and γ is a real-valued function of n variables $\alpha_1, \alpha_2, \ldots, \alpha_n$ in the domain A and k variables $i_1, i_2, \ldots, i_k$ in the domain I (I being the union of all sets of attribute domains). We may have $\gamma = a_1$, $\gamma = i_2$ or $\gamma = 2a_3 + a_2 + i_4 - 5$.

$$\phi = < \alpha_1\theta\gamma_1 > \wedge \ldots \wedge\ < \alpha_{m-1}\theta\gamma_{m-1} > \wedge < \alpha_m\theta\gamma_m > \tag{6.15}$$

Definition 6.16. *Feedback Proposal Classification*
A feedback proposal classification formula classifies each attribute j of the proposals that lost a round allowing an *OA* to provide qualitative feedback according to definition 6.5. Considering that $O^t_{r\to o}[j]$ represents the attribute j of a proposal, from r to o at round t, that lost a round, a Fc_o formula should be defined by each *OA* considering the type of environment (cooperative, competitive or mixed), allowing to calculate a variation of j in relation to a value of reference, i.e,

$$\Delta_j = Fc_o\left(O^t_{r\to o}[j]\right) \tag{6.16}$$

Additionally, a γ value that allows the definition of a range to classify the Δ_j in one of the values of set F should be defined (Definition 6.5).

6.3.3 Related to the Q-Learning Algorithm

Definition 6.17. *State*
A state ST in the q-learning algorithm used by a respondent agent r is represented by a n-tuple that includes an optional n-tuple composed by relevant attributes pa from the set of problem domain attribute PA and the feedback f received for each attribute from the organizer agent:

$$ST^r = \langle [\langle pa_1, \ldots, pa_w \rangle], f_1, \ldots, f_i \rangle \ with \ f_i \in F \tag{6.17}$$

i is the number of attributes.

Definition 6.18. *Action*
An action AT in the q-learning algorithm used by a respondent agent r is represented by a n-tuple composed of actions to be applied to each of the attributes of the problem, plus an optional problem domain action da defined by the system designer:

$$AT^r = \left\langle \langle [da_1], at_1, \ldots, at_m \rangle_1, \ldots, \langle [da_{i-1}], at_1, \ldots, at_m \rangle_{i-1}, \langle [da_i], at_1, \ldots, at_m \rangle_i \right\rangle \tag{6.18}$$

$$with \ at_k \in ACT$$

where m is the number of attributes of a dimension and i the number of dimensions. Please note that the members of the set ACT can be defined according to specific application domain needs. However, it should include at least those members that express increase, decrease and keep actions, so that the q-learning algorithm can work properly. For example, $ACT = \{increase, decrease, keep\}$.

Definition 6.19. *Reward of a Proposal*
A proposal reward rw in the q-learning algorithm used by a respondent agent r is a value calculated after receiving the feedback from the organizer agent o on the proposal presented in round t. A reward function Rw needs to be defined that should include a penalization when the respondent agent loses the round:

$$rw = Rw_r^{t+1}\left(O_{r \to o}^t\right) \tag{6.19}$$

6.3.4 Related to the Domain Language

Definition 6.20. *call-for-proposal*
An action to initiate a negotiation process by making a call for proposals. It is a n-tuple that includes the sender agent o, the receiver agent r (belonging to the set of organizer agents O and respondent agents R, respectively), the cfp id, the problem to be solved P and the negotiation deadline nd:

$$cfp = \langle o, r, id, P, nd \rangle \ with \ o \in O, r \in R \ and \ nd, id \in \aleph \tag{6.20}$$

Definition 6.21. *propose*
An answer to a call-for-proposal or an existing proposal during a negotiation process. It is a n-tuple that includes the sender agent r, the receiver agent o (belonging to the set of respondent agents R and organizer agents O, respectively), the propose id and the solution S to the problem to be solved:

$$propose = \langle r, o, id, S \rangle \ with \ o \in O, r \in R, id \in \aleph \tag{6.21}$$

Definition 6.22. *accept-proposal*
An acceptance of a proposal that was previously submitted. It is a n-tuple that includes the sender agent o, the receiver agent r (belonging to the set of organizer agents O and respondent agents R, respectively) and the id of the accepted proposal:

$$accept\_proposal = \langle o, r, id \rangle \ with\ o \in O, r \in R, id \in \aleph \tag{6.22}$$

Definition 6.23. *reject-proposal*
The action of rejecting a proposal that was previously submitted and, optionally, give feedback. It is a n-tuple that includes the sender agent o, the receiver agent r (belonging to the set of organizer agents O and respondent agents R, respectively), the id of the rejected proposal and (optionally) a n-tuple of elements of the feedback set F for all problem dimensions i included in the proposal:

$$reject\_proposal = \langle o, r, id, [\langle f_1, \ldots, f_i \rangle] \rangle \ with\ o \in O, r \in R, id \in \aleph, f_i \in F \tag{6.23}$$

Definition 6.24. *failure*
An information from the sender agent a to receiver agent b communicating that due to reason θ (a proposition) it was not possible to present a solution to the problem P. It is a n-tuple that includes the sender agent a, the receiver agent b, the problem P to be solved and a reason θ for failure:

$$failure = \langle a, b, P, \theta \rangle \tag{6.24}$$

Definition 6.25. *inform*
An information from the sender agent a to receiver agent b that a proposition θ is true. It is a n-tuple that includes the sender agent a, the receiver agent b and the proposition θ:

$$inform = \langle a, b, \theta \rangle \tag{6.25}$$

Definition 6.26. *refuse*
The action of refusing to present a solution to a problem P and explaining the reason for the refusal through proposition θ. It is a n-tuple that includes the sender agent a, the receiver agent b, the problem P and the proposition θ with the reason:

$$refuse = \langle a, b, P, \theta \rangle \tag{6.26}$$

Definition 6.27. *request*
Agent a requests agent b to execute statement θ (a proposition) and (optionally) complying with restrictions RT. It is a n-tuple that includes the sender agent a, the receiver agent b, the statement θ and (optionally) the set of restrictions RT:

$$request = \langle a, b, \theta, [RT] \rangle \tag{6.27}$$

6.4 Communication and Domain Language

The objective of a *Communication Language* is to facilitate the communication within agent's interactions. Two major proposals have been advanced, namely the *Knowledge Query and Ma-*

nipulation Language (Finin *et al.*, 1994) and *Foundation for Intelligent Physical Agents' Agent Communication Language* (FIPA-ACL) standard (FIPA, 2002a). We have adopted the latter. Usually, the elements of the communication language are known as speech acts, utterances or locutions (Searle, 1969; Traum, 1999). FIPA-ACL contains 22 locutions and the ones we use on the GQN protocol are defined in Table 6.1.

Table 6.1 FIPA-ACL Locutions used in GQN

Locution	**Example**	
Accept-proposal	$accept\_proposal = \langle o, r, id \rangle$	Definition 6.22
Call-for-proposal (cfp)	$cfp = \langle o, r, id, P, nd \rangle$	Definition 6.20
Failure	$failure = \langle a, b, P, \theta \rangle$	Definition 6.24
Inform	$inform = \langle a, b, \theta \rangle$	Definition 6.25
Propose	$propose = \langle r, o, id, S \rangle$	Definition 6.21
Refuse	$refuse = \langle a, b, P, \theta \rangle$	Definition 6.26
Reject-proposal	$reject\_proposal = \langle o, r, id, [\langle f_1, \ldots, f_i \rangle] \rangle$	Definition 6.23
Request	$request = \langle a, b, \theta, [RT] \rangle$	Definition 6.27

A *Domain Language* is necessary so that the agents are able to refer and understand the concepts of the domain, proposals, time and, of course, the object of negotiation. This includes the attributes under negotiation as well as constants that represent the negotiation attributes' value and any other needed symbols. Efforts have been made to provide standard domain languages that can be used by agents in heterogeneous environments. For example, *DARPA Agent Markup Language* (Hendler & McGuiness, 2000) and *W3C Web Ontology Language* (OWL) (McGuiness & van Harmelen, 2003). In our proposed protocol, the kind of domain language used is decided by the system designer. It is possible to use one of the standards or define another domain language using an *Object Oriented* language or any other. Independently of the domain language used, at least the concepts described in section 6.3 from 6.1 to 6.19 need to be defined.

6.5 Interaction Protocol

The GQN Protocol is designed for competitive, cooperative or mixed environments, meaning that it supports agents with expertise to present a full solution to a problem, as well as agents that alone might not be able to present a full solution to a problem because they do not have the full expertise to do it. Because they lack that expertise they need the cooperation of other agents (experts on subparts of a problem) to be able to reach a full solution. According to our definition of negotiation model (Definition 6.6), in GQN the agents can assume two different types: Organizer and Respondent. The organizer ones are responsible for defining what a problem is and to divide it in sub-problems (we call it dimensions of the problem - see definition 6.1 and 6.2). They are also responsible for initiating the negotiation and to evaluate the proposals received from the respondent ones. The respondent agents have an expertise (announced as one or more competences according to definition 6.11) on one or more dimensions (subparts of the problem) and are entitled to negotiate with other respondent agents (if they do not have the full expertise) to be able to present proposals for the whole problem to the organizer agent. It is possible to have more than one respondent agent with expertise in the same dimension of a problem (e.g., although they have the same expertise they might use a different problem solving method/strategy).

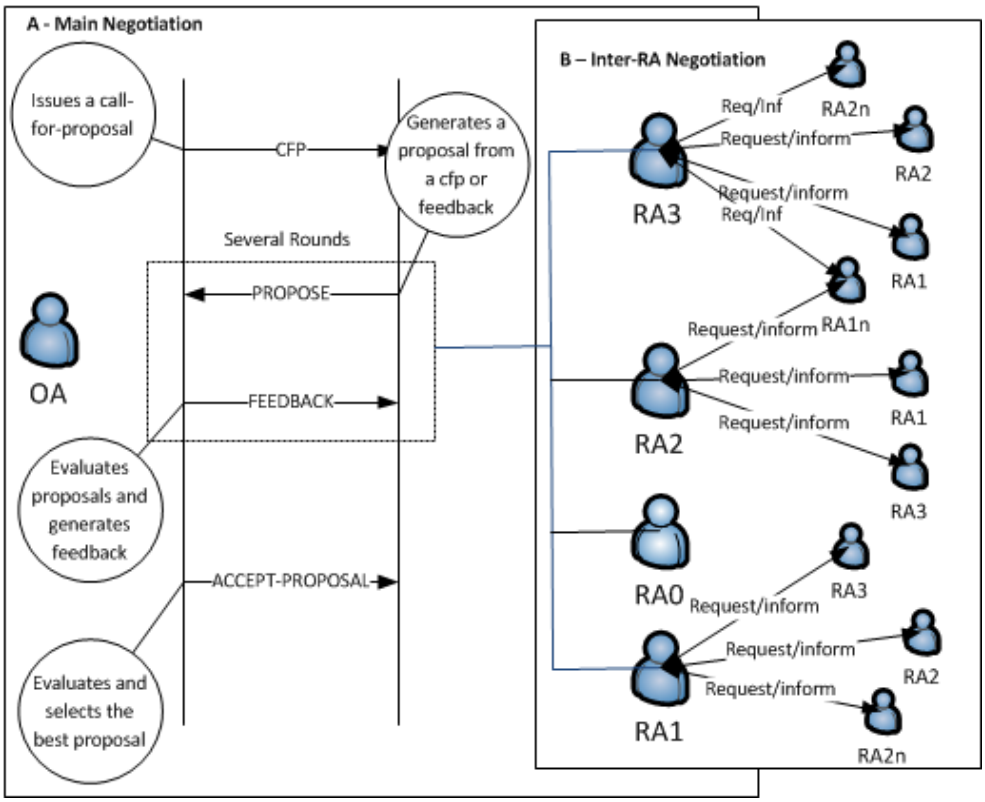


Fig. 6.1 GQN Agent Types, Roles and Negotiations

Looking at Figure 6.1 we can see that the agents can participate in several negotiations and assume different roles. The main negotiation is initiated by the organizer agent *OA* and includes as participants the respondent agents *RA0*, *RA1*, *RA2* and *RA3*. The solution to the problem (see definition 6.4 and 6.3) will be the outcome of this main negotiation. If the respondent agent has the full expertise to present a proposal, it does not need to engagage in a negotiation process with other respondent agents (it is the case of agent *RA0*). In the example of Figure 6.1 agents *RA1*, *RA2* and *RA3* do not possess the full expertise so, they will not be able to present a proposal with a full solution to the problem. Agent *RA3* initiates an *Inter-RA negotiation* with respondent agents *RA2*, *RA1*, *RA2n* and *RA1n* as participants. The *RA1n* and *RA2n* have the same expertise as *RA1* and *RA2*, respectively, and might participate in the main negotiation if they wish. The outcome will be a proposal that agent *RA3* will submit in a specific round of the main negotiation. Likewise, agent *RA2* initiates a negotiation with agent *RA1* and *RA3* and agent *RA1* initiates a negotiation with agents *RA2* and *RA3*.

As a summary, in Figure 6.1 we have:

- One *main negotiation* that has several rounds and negotiations between respondent agents (*inter-RA*) in each round.
- The *OA* agent assumes the role of *initiator* on the main negotiation and agents *RA0*, *RA1*, *RA2* and *RA3* the role of *participant*.
- Agents *RA1*, *RA2* and *RA3* are *initiators* and *participants* in the inter-RA negotiation.

An interaction protocol specifies who is allowed to say what, at each stage of the negotiation process. In Figure 6.2 we have a state diagram for the main negotiation, that is, the negotiation between the organizer agent *OA* and the respondent agents *RA0*, *RA1*, *RA2* and *RA3* (to make the

diagram simpler we call them *RAs*). After defining what a problem is and its dimensions the *OA* issues a call-for-proposal using the performative `cfp(oa,RAs,id,P,nd)` (Definition 6.20) to the *RAs* involved. As expected, the preferred values of the *OA* are not included in the `cfp`. Because we have three *RAs* (*RA1*, *RA2* and *RA3*) with different expertises, it means that our problem has three dimensions: one that corresponds to the expertise (announced as a competence - definition 6.11) of each one of them. The agent *RA0* has competences in *all* the three dimensions. The solution to the problem will be the aggregation of all dimensions. The issue of a `cfp` leads the negotiation to **state 1** and to the first round of the negotiation.

After **state 1** it is possible to reach three different states. If only one of the respondent agents present a proposal through performative `propose(ra,oa,id,S)` (Definition 6.21) and the others refuse to present a proposal (through performative `refuse(RAs,oa,P,`θ`)` - see definition 6.26) the negotiation enters **state 3**. Remember that each *RA* needs to present a proposal with the solution for the whole problem and not only for the dimension that corresponds to its competence. We will explain later how each *RA* is able to do that. After **state 3** the negotiation might end according to the *Termination Rule* (Section 6.6). If the proposal is accepted by the *OA* according to the *outcome determination rule*, then an `accept-proposal(oa,RA,id)` (Definition 6.22) is issued and the negotiation reaches the **final state**. If the proposal is not accepted, the *OA* changes its preferences and issues a new `cfp` leading the negotiation to **state 1**. If, after **state 1**, none of the respondent agents wants to present a proposal, they issue a refuse performative. This will lead the negotiation to **state 4**. In **state 4** we do not have any solution to our problem. In this case, the round ends and the *OA* has the same two options mentioned before: (1) change its preferences, extend the *negotiation deadline* and start the negotiation by issuing a new call-for-proposal; or (2) end the negotiation without a solution.

In the latter case the negotiation reaches the **final state**. In the former, a new round starts and the negotiation enters **state 1** again. The decision to extend the *negotiation deadline* and change its preferences, depends on the strategy adopted by the *OA*.

Finally, from **state 1** and if all respondent agents present a proposal (through performative `propose`), we reach **state 2**. At this time the *OA* evaluates all proposals using an evaluation function and chooses the winner of the round (Definition 6.13). If the *termination and outcome rule* apply, then the negotiation goes to the **final state** with the acceptance of the winner proposal through a `accept-proposal` message. The agents that lost receive a `reject-proposal` message. If the rules do not apply, the *OA* sends an `inform(oa,RA,round_winner)` (Definition 6.25) to the agent that presented the winner proposal and to all others a `reject-proposal(oa,RAs, id,feedb)` (Definition 6.23). In the reject-proposal message the *OA* includes qualitative feedback for each attribute of the received proposal. Each attribute value is classified according to the preferences of the *OA* (see Definition 6.16) and that classification is sent as feedback. With these actions, the negotiation goes to **state 5** and the round ends. After **state 5**, a new round starts. The winner agent of the previous round presents the same proposal. The others have two choices: make a new proposal improving the one sent in the previous round (using the feedback provided) or refusing to present a new proposal. In the latter case, the negotiation goes to **state 3**. In the former case, the negotiation goes to **state 2** and the proposal evaluation and feedback generation are performed for this round. After any of the termination rules are valid (for example, reaching a negotiation deadline or if the evaluation of a proposal is considered good enough by the *OA*) the *OA* ends the negotiation accepting the best proposal until that moment and rejecting all the others.

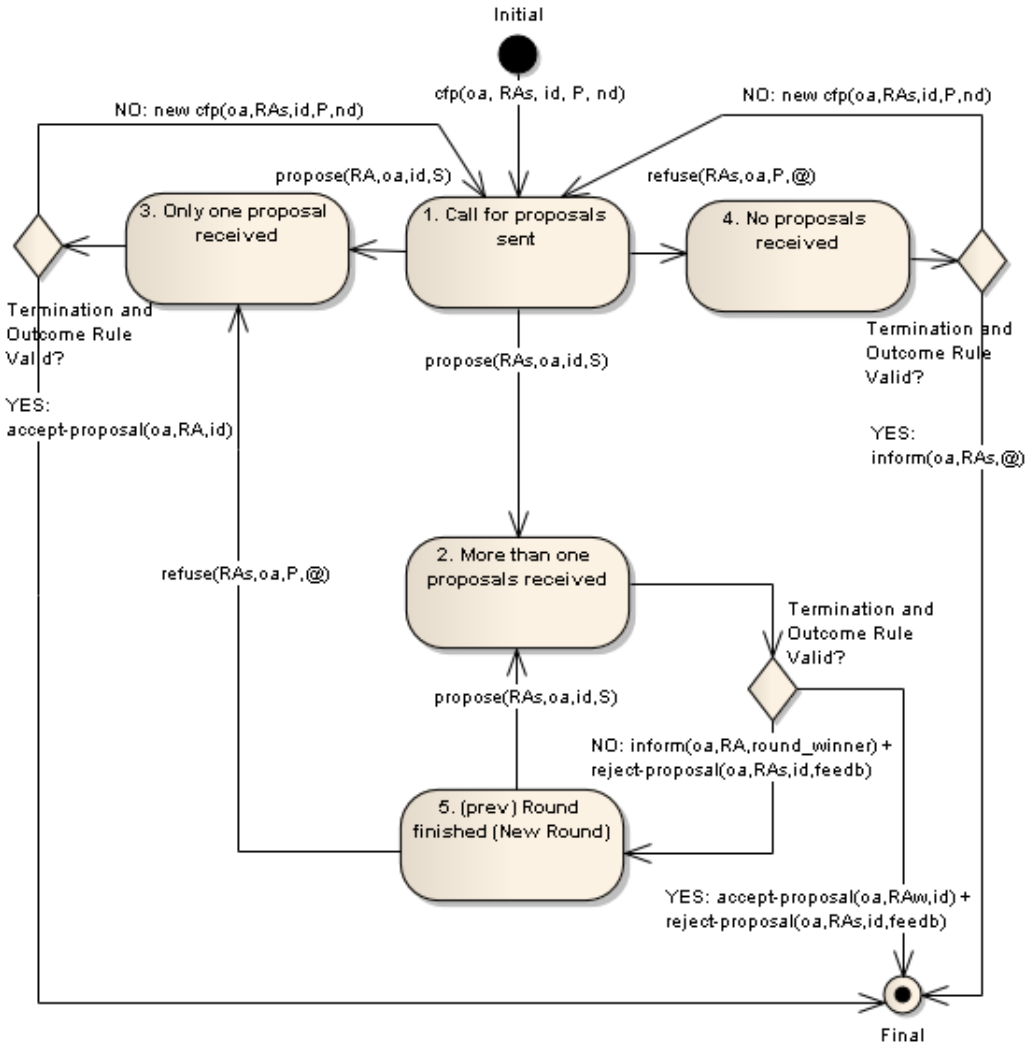


Fig. 6.2 Main Negotiation State Diagram

The state diagram of Figure 6.2 does not show the negotiation that happens between the respondent agents. Remember that each *RA* might not have the necessary expertise (announced as a competence) to present a proposal that includes all the dimensions of the problem and, as such, a full solution. In that case, he will have to negotiate with the other *RAs*.

Figure 6.3 shows the state diagram for the inter-RA negotiation. In the diagram we represent a single agent as *ra* and all the others (excluding the *ra*) as *RAs*. Each respondent agent has a different competence. Please note that, for simplicity, we did not include agents with the same expertise for the same dimension. However and when necessary we will explain how the protocol works in this scenario. As we stated in the beginning, our protocol can be used in a cooperative, competitive and/or mixed environment meaning that the agents might have some degree of self-interest. Each respondent agent is interested in proposing the best candidate solution that gives the higher utility (see definition 6.12) for the dimension on which he is an expert. So, in the example of Figure 6.3, the *ra* starts by getting the candidate solutions for the dimension on which he is an expert and orders them in descendent order of utility. Let the set of possible solutions found by *ra* be called *CS* and cs_i an element of this set.

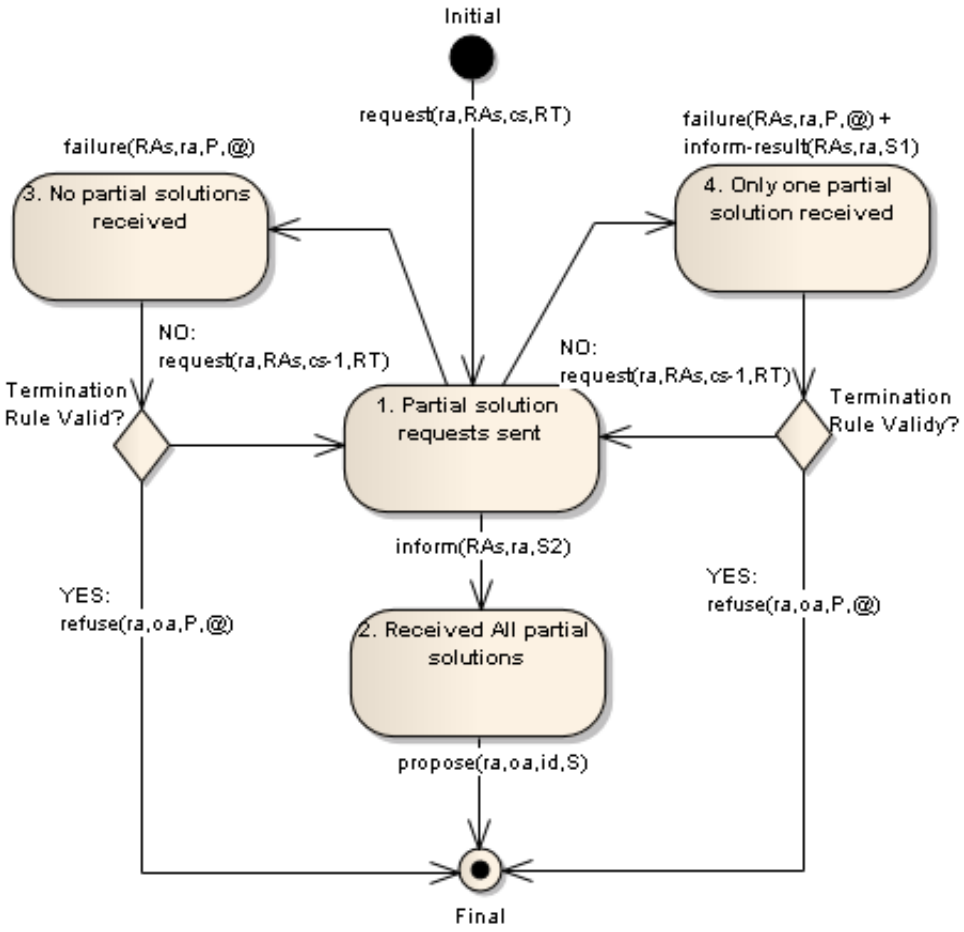


Fig. 6.3 Inter-RA Negotiation State Diagram

Starting with the first possible solution from *CS* the *ra* issues a simultaneous request to all the other *RAs* using performative request: `request(ra,RAs,cs,RT)` (Definition 6.27) asking for partial solutions from the other *RAs* that can complete the candidate solution he found for its dimension. These requests includes the set *RT* that corresponds to the restrictions that each respondent agent needs to comply when sending the partial solutions (Definition 6.15). The restrictions sent to each of the respondent agents can be different. These restrictions guarantee that the interdependencies between the partial solutions are solved, otherwise the *ra* could end up with an invalid candidate solution (see definition 6.14). The issue of the requests leads this negotiation to **state 1**.

From **state 1** there are three possible states that can be achieved. **State 4** corresponds to the state where only one partial-solution of a single dimension is received. The *RAs* might not have a partial-solution that complies with the request sent by *ra*. In this case, they will send a failure performative. For example, `failure(RAs,ra,P,`θ`)` (Definition 6.24). The statement θ includes the reason for not presenting the partial-solution. The *RAs* that have a partial-solution, will send it through a `inform-result` message. **State 3** corresponds to the state where none of the *Ras* sent a partial-solution. In the end of **state 3** and **4** the negotiation *termination rule* is checked. If valid, the inter-RA negotiation goes to the **final state** and the *ra* issues a `refuse` message to the *OA*, meaning that it will not make a proposal on that round of the main negotiation. If the rule is not valid, then the *ra* uses the next element of the set *CS* of candidate solutions (in the diagram it is

represented by *CS-1*) to issue a new request to each of the *RAs* (probably with different restrictions because the candidate solution is different), and the negotiation goes to **state 1** again.

Finally, from **state 1**, the other state that can be achieved is **state 2** and corresponds to the state where all *RAs* presented a partial-solution to *ra* that covers all the dimensions. The *RAs* do that through a `inform(RAs,ra,S2)` message. If there is more than one agent with the same expertise for the same dimension, *ra* will have to choose from the combination of partial solutions, the one that gives him more utility, according to the *outcome determination rule*. For example, let us suppose that RA_4 and RA_5 have the same expertise than RA_2 and RA_3, respectively, and all of them presented a partial-solution. In this case, *ra* will have to consider all the aggregate combinations of candidate partial solutions (e.g., ra+ra_2+ra_3, ra+ra_2+ra_4, ra+ra_4+ra_3, etc.), calculate the utility of each one and choose the one with the highest utility. Having its own CS_i plus the best aggregate combination of partial solutions the *ra* ends the negotiation by issuing a `propose(ra,oa,id,S)` to the *OA*.

6.6 Rules of Interaction

Before starting to explain the internal features of the agents, we just need to indicate more explicitly the rules of interaction used by our protocol, namely: rules for admission, participant withdrawal, negotiation termination, proposal validity, outcome determination and commitment rules.

Rule 1: The *admission rule* specifies when and under what conditions an agent can participate in a negotiation. For the respondent agents to participate in a negotiation, they should have one or more competences (Definition 6.11) that corresponds to one or more of the dimensions (Definition 6.2) of the problem.

Rule 2: The *participant withdrawal rule* specifies when and under what conditions a participant may withdraw from the negotiation. The respondent agents can withdraw from the main negotiation in the following conditions (see state diagram of Figure 6.2):

1. By refusing to present a proposal due to lack of competence. They do that through a *refuse* performative (transition from state 1 to state 4).
2. When they cannot find a solution to a problem. They do that through a *refuse* performative (transition from state 5 to state 3).

Regarding the Inter-RA negotiation (see state diagram of Figure 6.3) they can withdraw in the following condition:

3. When they cannot comply with restrictions. They do that through a *failure* performative (transition from state 1 to 3 and 1 to 4).

Rule 3: The *negotiation termination rule* specifies when and under what conditions the negotiation ends. The main negotiation ends (Figure 6.2) when any of the following conditions are met:

1. When only one proposal is presented in a round (transition from state 3 to the final state).
2. When the negotiation deadline reaches the value defined by the organizer agent (transition from state 2 to the final state).
3. If there are no proposals presented in the first round (transition from state 4 to the final state).

Regarding conditions 1 and 2, if the evaluation of the proposal is within a pre-defined range, the proposal is accepted and the negotiation ends with success. If the evaluation of the proposal is outside that range, the proposal is rejected. Being rejected or in the case of condition 3, the *OA* has two options: (1) to extend the negotiation deadline, change the preferences and start a new negotiation by issuing a new call-for-proposal or, (2) to end the negotiation without a solution. Regarding the first option, there is a finite temporal limit to this extension defined by the *OA* according to its strategy.

Regarding the negotiation inter-respondent agents (Figure 6.3) it ends when any of the following conditions is valid.

4. Only one partial-solution is received from the participants and the initiator has no more candidate solutions available (transition from state 4 to the final state).
5. No partial solutions received from the participants and the initiator has no more candidate solutions available (transition from state 3 to the final state).
6. When all partial solutions are received (transition from state 2 to the final state).

Regarding the last condition, the initiator chooses the aggregate combination of partial solutions that gives him the highest utility.

Rule 4: The *proposal validity rule* specifies when and under what conditions a proposal is valid. A proposal is valid when:

1. Includes a solution *S* (Definition 6.4) to the problem *P* (Definition 6.1) received in the *cfp*.
2. The rule 4.1 is valid and *S* is a coherent solution according to Definition 6.14.
3. The rule 4.1 is valid and the proposal was not previously submitted by an agent that lost the previous round.
4. When it is sent by the agent that won the previous round.

Rule 5: The *outcome determination rule* specifies the outcome of the interaction (for example, how to determine a winning proposal). A solution to a proposal is obtained when:

1. Regarding rule 3.1 and 3.2, if the evaluation of the proposal is within a pre-defined range, the proposal is accepted as an outcome of the negotiation.
2. Regarding rule 3.6, the aggregate combination that gives the highest utility to the initiator.

Rule 6: The *commitment rule* specifies how the commitments of agents should be managed (whether and when an agent can withdraw a previous commitment, etc.). The agents should follow or assume the following rules:

1. If an agreement is reached, that is, if there is a winner proposal, all parties involved will honor it.
2. There are no long-term commitments, that is, each negotiation stands alone and an agent cannot commit itself to any other future action than the one agreed-upon.

6.7 Organizer Agent Architecture

After having explained the external features of our protocol, it is time to explain the internal features, that is, the agent's architecture, decision mechanisms and knowledge bases, amongst others. We start by explaining the organizer agent architecture.

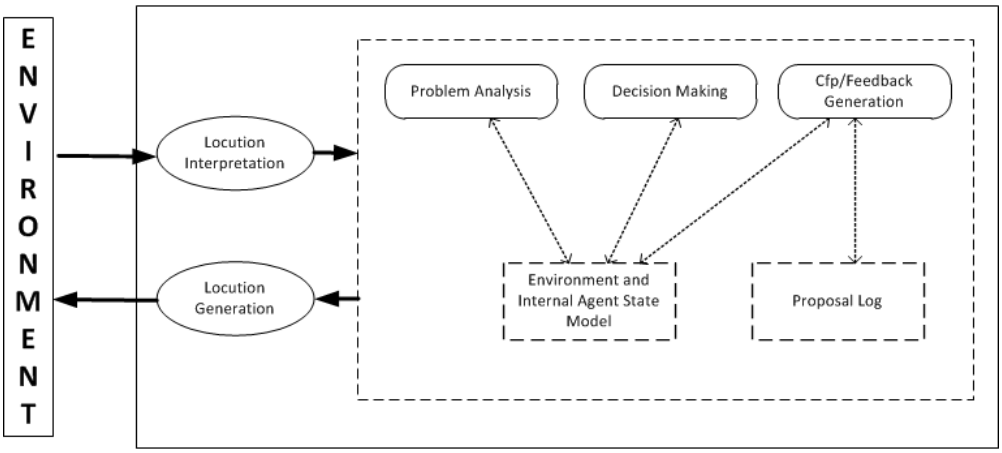


Fig. 6.4 Organizer Agent Architecture

As stated in Section 6.5 there are two types of agents: *Organizer* (OA) and *Respondent* (RA) agents. The *OAs* are responsible for defining what a problem is and to divide it in sub-problems if necessary. They are also responsible for the initiation of the negotiation and the evaluation of the proposals received from the respondent ones. The *RAs* have an expertise (defined by one or more competences) on a (or all) subpart of the problem (dimensions) and are responsible for presenting proposals to the *OA*. Although they might only have expertise in a single dimension, *RAs* propose to the *OA* a solution for the whole problem, which may imply the negotiation with others *RAs*. An *OA* views the negotiation process $NegP^o$ according to the structure represented by definition 6.7. Figure 6.4 shows the main components of an *organizer agent*. The role of each component is presented in the following sub-sections.

6.7.1 Locution Interpretation

This component is responsible for parsing the incoming messages. These messages include the locutions `request` from an external agent *(EA)* that is responsible for detecting the problem (this agent is not represented in the state diagram of Figure 6.2) and the locutions `propose` and `refuse` received from the respondent agents (see state diagram in Figure 6.2) and, in their content, concepts represented in the domain language.

6.7.2 Environment and Internal State Model

The environment includes the resources (e.g., variables or tuples) that are available for the agent for sensing (e.g., reading their values), for effecting (e.g., changing their value) or for consuming (e.g., extracting them from the environment). The internal agent state model includes the preferred attribute values, the attribute domain and the scoring functions, i.e., the set *VP*, set *I* and set *V*, respectively, of each dimension according to definition 6.2. It includes also the proposal classification function for the feedback generation (Definition 6.16). With this component, the agent is able to build and keep a model of the environment and of its internal state. It is a very important com-

ponent since almost all the other components will use it. It will help in the problem analysis (e.g., in defining the preferences for each attribute), in decision making (e.g., evaluation of proposals) and in the cfp/feedback generation (e.g., verifying if proposals are feasible and do not conflict with current observations of the environment and in generating feedback according to the classification of a proposal). We do not propose a specific way to build and specify the model. The choice of the best way to specify this model is close related to the choice of the domain language and, that choice, might constraint or impose some rules regarding the way the model is built.

6.7.3 Proposal Log

In this component, the proposals as well as the rejections exchanged during the negotiation rounds are stored. This will provide valuable information for the agent to make the best decisions regarding proposal and feedback generation. This log will also allow the agent to fulfill one of its responsibilities regarding the proposal validity rule by not allowing a respondent agent to present a previous proposal (rule 4). The proposal log L^o of an *OA* has the structure defined according to definition 6.9.

6.7.4 Problem Analysis

This component is responsible for defining the problem and for its decomposition. The strategy to define the number of dimensions as well as the number and attributes that characterize each dimension, is up to the agent designer. The protocol does not impose a specific strategy. It is possible to use a static or dynamic problem definition and decomposition approach. Regardless of the strategy used, it is mandatory that the problem be *decomposed with the dependencies (if any) defined through a restriction set*, to *avoid conflicts and incoherent solutions*. For each attribute j of the set A^i and for all dimensions i of the problem P the following needs to be defined:

- A range $[min^i_j, max^i_j]$ if the attribute domain is continuous or, if the domain is discrete, a list of the acceptable values (e.g., red, blue, yellow).
- A preferable value within the range defined above (if continuous) or that belongs to the list (if discrete).
- A scoring function $V^i_j : [min^i_j, max^i_j] \rightarrow [0,1]$ to score the value of issue j in the range of its acceptable values. For easier comparisons, the scores are kept in the interval [0, 1].
- An optional restriction set RT^i of restrictions (definition 6.15) that define the interdependencies between dimensions.

For each and all dimensions of the problem we need to define the relative importance of each attribute under negotiation as well as the relative importance of each dimension. We call $V^i\left(ps^i\right)$ to the score function of a partial-solution for dimension i and $V^P(S)$ to the score function of a solution. Let w^i_j be the importance of attribute j in dimension i. We assume the weights are normalized, i.e., $\sum_{j=1}^{m^i} w^i_j = 1$ for all $i \in P$ and m^i as the number of attributes in dimension i. For a partial-solution ps^i we define the score function as:

$$V^i\left(ps^i\right) = \sum_{j=1}^{m^i} w_j^i V_j^i\left(ps_j\right) \tag{6.28}$$

Finally, it is necessary to define the relative importance of each dimension in the problem. Let α_i be the importance of dimension i. We assume the weights are normalized, i.e., $\sum_{i=1}^{|P|} \alpha_i = 1$. For a solution S of a problem P we define the score function as:

$$V^p\left(S\right) = \sum_{i=1}^{|P|} \alpha_i V^i\left(ps_i\right) \tag{6.29}$$

Having this score function the utility of a proposal for an organizer agent is defined according to definition 6.12.

The *Internal Agent State Model* is updated with this information so that the other components will have the necessary knowledge to perform their roles. After the problem decomposition and the definition of the dimensions, the *Cfp/Feedback Generation* component will be able to generate the `cfp` and start the negotiation.

6.7.5 Decision Making

In each negotiation round, the *OA* is expected to receive a propose from each of the *RAs* participating in the negotiation that includes a solution S. This component is responsible for *evaluating each proposal* by calculating the utility of each one according to definition 6.12 and *deciding the round winner* according to definition 6.13. For the agents that lost the round, this component will have to *classify the proposals* according to a *Feedback Proposal Classification* formula (Definition 6.16). This classification will be used by a different component (i.e., *Cfp/Feedback Generation*) to inform the agents that they lost the round, so that those agents can present new proposals in a subsequent round (if they decide to do it). This process of alternating-proposals with feedback will end according to the termination rules (Section 6.6, rule 5). It is the responsibility of the decision making component to *select the negotiation winner* according to the outcome determination rule (Section 6.6, rule 5). Finally, this component is also responsible to *enforce and validate the admission rule* (rule 1) and the *proposal validity rule* (rule 4.1 and 4.3).

6.7.6 Cfp/Feedback Generation

Considering the analysis of the problem and the decision making that took place, this component is responsible for generating the call-for-proposals (when starting the negotiation that corresponds to state **initial state** in the diagram of Figure 6.2 or the feedback, after receiving proposals from the respondent agents (**state 2** in the same diagram).

In the former case, it has to define in the domain language the problem as created and decomposed in the *Problem Analysis* component and using the information available in the *Internal Agent State Model*. All this should be part of the `cfp` performative.

In the latter case, it should prepare and define in the domain language, the content to be included in the `inform` performative that will be sent to the agent that presented the round winner

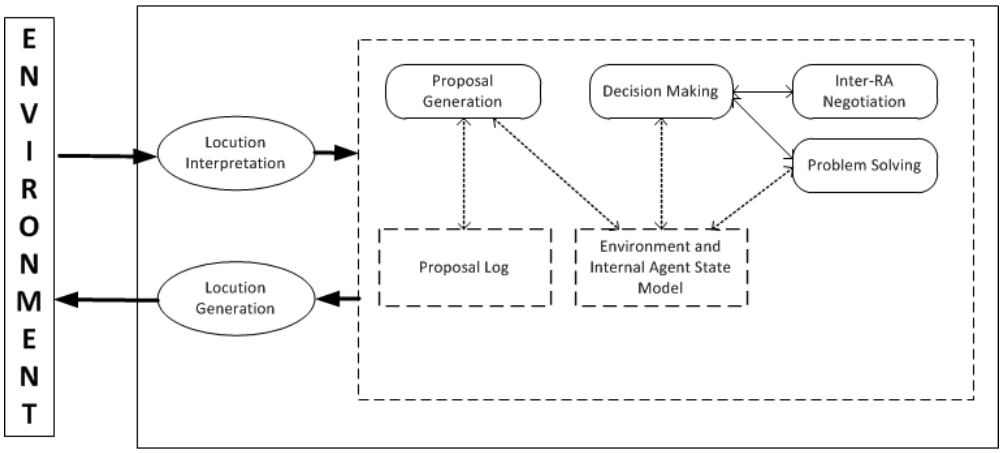


Fig. 6.5 Respondent Agent Architecture

proposal. Likewise, it should prepare the content and corresponding feedback, using the information prepared by the *Decision Making* component, to be sent to each of the agents that presented a proposal that lost, using a `reject-proposal` performative.

6.7.7 Locution Generation

This component is responsible for parsing the outgoing messages in the format required by the communication and domain language and according to the decisions that the agent has made. The possible locutions included in the messages are (see Figure 6.2): `cfp` (when starting a negotiation), `accept-proposal` (at the end of the negotiation and when there is a winner proposal), `reject-proposal` (during the negotiation rounds to reject proposals that did not win the round) and `inform` (to inform the winner respondent agent in the round).

6.8 Respondent Agent Architecture

As stated before, the respondent agents (*RAs*) have an expertise (announced as one or more competences according to definition 6.11) on a subpart of the problem (dimension) or, if that is the case, on all subparts of the problem, and are responsible for presenting proposals to the *OA*. An *RA* views the negotiation process $NegP^{Ra}$ according to the structure represented by definition 6.8. Figure 6.5 shows the main components of a respondent agent. The role of each component is presented in the following sub-sections.

6.8.1 Locution Interpretation

This component is responsible for parsing the incoming messages. Since *RAs* might assume two roles, i.e., participants in the negotiation between the *OA* and the *RAs* (main negotiation in Figure 6.1) and in inter-respondent agents negotiation (inter-RA negotiation in Figure 6.1),

and initiators in the negotiation between *RAs*, the messages might include the locutions: `cfp`, `accept-proposal`, `reject-proposal`, `inform`, `refuse`, `propose`, `request` and `failure` and, in their content, concepts represented in the domain language.

6.8.2 Environment and Internal State Model

As in the *OA* equivalent component, the environment includes the resources (e.g., variables or tuples) that are available for the agent for sensing (e.g., reading their values), for effecting (e.g., changing their value) or for consuming (e.g., extracting them from the environment). The internal agent state model includes the attribute names and domains as well as the competence scoring function, i.e., the set *CA*, set *CD* and scoring function *CV*, respectively according to definition 6.11. It includes also several elements needed for the Q-Learning algorithm, namely: a reward function *Rw*, a state *ST* and an action *AT* representation (see Definition 6.19, 6.17 and 6.18, respectively). Finally, it is also possible to include other information related to problem solving (e.g., objective function, etc.) if, besides Q-Learning, the designer wants to use an additional strategy. As in the *OA*, with this component, the agent is able to build and keep a model of the environment and of its internal state. It is a very important component since almost all the other components will use it. It will help in the proposal generation (e.g., generating them according to the competence and strategy), in decision making (e.g., selecting best proposals), in inter-respondent agents negotiation (e.g., generating requests according to preferences, evaluating responses and in guarantying the coherence of the solutions) and in problem solving (e.g., assuming the right space and action representation). As we state before, we do not propose a specific way to build and specify the model. The choice of the best way to specify this model is close related to the choice of the domain language and, that choice, might constraint or impose some rules regarding the way the model is built.

6.8.3 Proposal Log

The goal of this component is to record the generated proposals and the feedback received during the main negotiation rounds. It will also keep the requests and responses received during the inter-respondent agents negotiations. This will provide valuable information for the agent to make the best decisions regarding proposal and requests generation. The proposal log L^{Ra} of a *RA* has the structure defined according to definition 6.10.

6.8.4 Decision Making

During the main negotiation (Figure 6.2) and after receiving a `cfp` performative or a reject-proposal the respondent agent (*RA*) needs to decide what proposal to send on the next round or, if that is the case, withdraw from the negotiation according to the *participant withdrawal Rule* (rule 2). Remember that in the case of being declared as a round winner the *RA* will resend the winner proposal on the next round.

After receiving a *cfp* (in the first round) the *RA* needs to verify if it has competences to send a proposal. For that, this component has to *validate the Admission Rule* (Rule 1). Since we are in the first round and considering that the organizer agent's preferences are private, the *RA* cannot infer those preferences. Because of that, it will have to send a proposal according to its own preferences, complying with the attributes domain received as part of the problem (set I according to definition 6.2) and complementing the proposal with the negotiation with other respondent agents, if necessary. Remember that an *RA* might have competences to present a solution to the whole problem or only to a subpart of the problem (i.e., a partial-solution that is an answer to a dimension of a problem according to definition 6.3).

After receiving from the *Problem Solving* component the list of candidate partial-solutions, this component will *evaluate them using the RA score function* and they will be ordered according to their utility. If the *RA* has competences to present a complete solution, then, it will use this ordered list to build its proposal, if it does not, then it will start the Inter-RA negotiation (through the *Inter-RA Negotiation* component) using the ordered list. It is the responsiblility of this component to *check the coherence of the partial-solutions* (Definition 6.14) received from the other RAs and to select the *winner of the Inter-RA negotiation* according to the outcome determination rule 5.2. The winner candidate solution will be the proposal presented by the *RA* through a `propose` performative in the main negotiation.

After the first round and in the case of receiving a `reject-proposal`, the *RA* needs to send a proposal not only according to its own preferences but, also, according to the feedback received from the *OA*.

Our protocol inherits the characteristics of the Q-Negotiation protocol (Rocha & Oliveira, 2001) and, as such, we can use the Q-Learning algorithm (Watkins & Dayan, 1992) during proposal formulation. This method of learning how to formulate new proposals is implemented in the *Problem Solving* component. The role of the *Decision Making* here is to *select the best candidate solution according to the action of the Q-Learning (Definition 6.18)*. This is done in five steps:

1. After receiving a list of candidate solutions from the *Problem Solving* component those that do not comply with the feedback received are removed;
2. If the strategy to be used is based only on the Q-Negotiation algorithm, then a list of actions with highest probability according to the Boltzmann exploration formula (Leslie Pack Kaelbling & Moore, 1996) is selected, i.e.,

$$p(a) = \frac{e^{Q(s,a)/t}}{\sum_{b \in AT} e^{Q(s,b)/t}} \; with \; t > 0 \; and \; b, a \in AT \; and \; s \in ST \tag{6.30}$$

   (The Q formula (Equation 6.31) is specific to the Q-Learning and the *Q-Value* associated to each $< state, action >$ pair, i.e., $Q(s,a)$ is updated every time a feedback is received after presenting a proposal. The Q formula as well as this process of updating the *Q-Value* is explained on the *Problem Solving* component);
3. For each action on this list a candidate solution is selected;
4. If the *RA* does not have competences to present a complete solution, an Inter-RA Negotiation is started using this new list;
5. the best candidate solution according to the outcome of this negotiation will be the proposal presented by the *RA* in the main negotiation.

This component is also responsible for *the enforcement and validation* of (Rule 3.4, 3.5 and 3.6).

6.8.5 Problem Solving

This component is responsible for implementing the algorithm or algorithms used in the problem solving strategy of the agent, i.e., in generating valid candidate solutions according to the agent's preferences (if it is a response to a `cfp`) or considering the feedback received from the organizer agent after presenting a proposal in a previous round. As stated before, our protocol inherits the characteristics of the Q-Negotiation protocol (Rocha & Oliveira, 2001). Additionally, each *RA* can implement a specific problem solving strategy. For example, in the *Disruption Management* application example in section 6.10.1 the Q-Learning algorithm is used to learn the best "domain operators" (e.g., exchange aircraft, use reserve crew, etc.). However, to look for candidate solutions according to the domain operator, a simulated annealing algorithm (Kirkpatrick *et al.*, 1983) is used. Due to the use of a learning algorithm, this component is also responsible for the adaptability characteristic of our protocol.

In this section we will not detail the Q-Negotiation algorithm or the Q-Learning algorithm. We are just going to detail what a system designer needs to define to be able to use it. The first thing to do is to define what a *state* and an *action* is. In our negotiation protocol the state is represented according to definition 6.17. The n-tuple ST^r is composed by the feedback received from the organizer agent for each attribute in the round. Additionally and optionally, it might include a n-tuple composed of attributes from the problem domain. If the strategy of the respondent agent in order to prepare a new proposal is based only on the q-learning algorithm, there is no need to include this additional n-tuple. However, when the q-learning algorithm is used as a complement of another strategy (as in the application example in section 6.10.1) then, it might make sense to include this n-tuple.

An action AT^r is represented as a n-tuple according to definition 6.18. This n-tuple indicates the possible actions to be taken over an attribute value considering the feedback received. These actions may vary according to the application domain needs. However, at least the following should be defined:

- **Increase** the value $O[j]$ of attribute j according to a strategy ϕ in round t+1:

$$O^{t+1}[j] = Inc\left(O^t[j], \phi_j\right)$$

- **Decrease** the value $O[j]$ of attribute j according to a strategy ϕ in round t+1:

$$O^{t+1}[j] = Dec\left(O^t[j], \phi_j\right)$$

- **Keep** the value $O[j]$ of attribute j in round t+1:

$$O^{t+1}[j] = O^t[j]$$

It is up to the system or agent designer to decide if each *RA* has a specific strategy ϕ for each attribute j or if it should use the same for all attributes, regarding the action to be performed. It can be as simple as increasing or decreasing a pre-defined value, i.e, $\phi_j = \Delta_j$ or choose the next value above (increase) or the next value below (decrease) that gives more utility to the agent or according to its order of preference. So, in round *t+1*, when the respondent agent r receives the feedback from the organizer agent o to the proposal O presented in the previous round t, it is

necessary to calculate the reward *rw* value according to a formula defined by the system designer, i.e., $Rw_r^{t+1}(O_{r\to o}^t)$ (see Definition 6.19).

According to the Q-Learning algorithm, when an agent is in state *st* and executes the action *at* receiving the reward *rw* and reaching the state st^* (by executing action *at* in state *st*) the corresponding Q-Value *Q(st,at)* is updated according to the formula

$$Q(st,at) = Q(st,at) + \alpha(rw + \gamma \max_a Q(st^*,a) - Q(st,at)) \tag{6.31}$$

where $0 \leq \alpha \leq 1$ and $0 \leq \gamma \leq 1$. a represents the set of admissible actions when in state st^*, α the learning rate and γ the discount factor. This formula is the standard formula used on the Q-Learning algorithm and the interested reader should see (Watkins & Dayan, 1992) for more details about it.

6.8.6 Inter-RA Negotiation

As we stated before, this protocol was designed for environments that support agents with expertise to present a full solution to a problem, as well as agents that might possess expertise only in a subpart of the problem. The set of respondent agents in a negotiation should include as many respondent agents as necessary with expertise that covers all dimensions of the problem. For example, in Figure 6.1 we have three respondent agents (*RA1*, *RA2* and *RA3*) one for each dimension of the problem plus two additional agents (*RA1n* and *RA2n*) that have similar expertise than two of the other agents (using a different problem solving strategy). Although not presented in Figure 6.1 (for simplicity) these agents also participate on the *main negotiation* together with agents *RA1*, *RA2* and *RA3*. Agent *R0* has expertise to present a full solution.

In a round each *RA* that participates in the *main negotiation* will present a proposal to the organizer agent that includes a complete solution. So, each *RA* after getting the best candidate solutions for its expertise will need to start a negotiation with the other respondent agents to be able to complete the proposal (the *RAs* that have competences to present a complete solution do not engage in this negotiation, as it is the case of *RA0*). Continuing with the example in Figure 6.1 the agent *RA3* assumes the role of organizer and *initiates* a negotiation with *RA1*, *RA2* and, in this particular case, with *RA1n* and *RA2n*. Likewise, *RA2 initiates* a negotiation with *RA3*, *RA1* and *RA1n* and, finally, *RA1* with *RA3*, *RA2* and *RA2n*. The *inter-RA Negotiation* component is responsible for managing this negotiation.

Figure 6.3 shows the state diagram for this *inter-RA negotiation* and an explanation of this diagram is given in section 6.5. In this section we will only point out the main characteristics of this negotiation, e.g.:

- To start an inter-RA negotiation the *RA* that assumes the role of *organizer* has a list of candidate solutions ordered according to its strategy;
- Starting with the first item of the list, send *requests* to the *RAs* including the necessary *restrictions* that guarantees that the interdependencies between partial solutions (dimensions) are solved. If there is a feedback to the previous proposal (received from the organizer agent in the main negotiation) it should be included in the *request*;
- If a partial-solution is not received from each of the *RAs* needed to complete a proposal then a new *request* is sent using the next item on the list;

- If, at least, a partial-solution is received from each of those agents, then the outcome is the aggregate combination of partial solutions with the highest utility (according to the *outcome determination rule* 5.2);
- The *inter-RA negotiation* ends according to the *termination rule* (rule 3.4-6)

6.8.7 Proposal Generation

The main responsibility of this component is to prepare the content of the `propose` performative that will be sent by the respondent agent to the organizer agent in each round. From the several partial-solutions received (one for each dimension of the problem) it will define in the domain language the n-tuple that represents a Solution according to definition 6.4. To be able to fulfill this responsibility, this component is dependent on the work performed by the *Problem Solving*, *Inter-RA Negotiation* and *Decision Making* components as well as on the information that is kept in the *Proposal Log* and in the *Environment and Internal Agent State Model*.

6.8.8 Locution Generation

This component is responsible for parsing the outgoing messages in the format required by the communication and domain language and according to the decisions that the agent has made. As stated in the *Locution Interpretation* component, the *RAs* have several roles, so, this component is able to include the same locutions in the outgoing messages as those possible for the incoming messages, i.e., `cfp`, `accept-proposal`, `reject-proposal`, `inform`, `refuse`, `propose`, `request` and `failure` and, in their content, concepts represented by the domain language.

6.9 Theoretical Analysis

The goal of this section is twofold: (1) to prove that our negotiation process will end after a finite number of steps. For that we will analyze the properties of the protocol, specially, the termination property (Rule 3 in Section 6.6); and (2) to prove that by the end of the negotiation we will have a solution that is better than the solution presented in the previous rounds (or, in the worst scenario, equal to), meaning that our protocol converges to better solutions. For that we are going to follow a methodology similar to the one used in (Dang & Huhns, 2005). Using the state diagrams presented in Figure 6.2 and in Figure 6.3 we can find the following property:

Property 1: Given our negotiation model (Definition 6.6) a negotiation process using the proposed protocol in this chapter ends after a finite number of steps.
Proof: Our negotiation model can be seen as several one-to-many negotiation occurring simultaneously, meaning that each negotiation process is independent. Figure 6.2 represents the state diagram of the main negotiation between one *OA* and several *RAs* and Figure 6.3 the state diagram of the inter-RA negotiation. In the first diagram we see three loops that can occur during the negotiation process: (1) a loop on state 1 and 4, i.e., the *OA* does not receive proposals from the *RAs*

and decides to extend or not the negotiation deadline; (2) a loop on state 1 and 3, similar to the previous ones but for the case of receiving one proposal; (3) a loop on states 2 and 5, i.e., the *RAs* and the *OA* keep exchanging counterproposals. In the second diagram we have two loops: (3) a loop on state 1 and 3, i.e., the initiator *RA* does not receive any partial solutions from the other *RAs* and relaxes its preferences; (4) a loop on state 1 and 4 that is very similar to the previous one but for the case of one proposal received. To prove *property 1* we must prove that there is no infinite sequence of loops on the above five loops.

1. In loop 1-4 in Figure 6.2, the *OA* is in a situation where none of the *RAs* presented a proposal. This situation can only occur in the first round because, after the first round and according to Definition 6.13, it is possible to have a round winner. In this case, it is up to the *OA* to decide if it changes its preferences and extends or not the negotiation deadline. If the decision is to not extend and accept the outcome then the negotiation ends. If the decision is to extend then, there is a finite temporal limit to that extension. The negotiation termination rule 3.3 and the outcome determination rule 5, guarantees this limit.
2. In loop 1-3, the *OA* is in a situation where only a proposal was received. As in the previous loop the negotiation termination rule and the outcome determination rule, ensures that this loop ends.
3. In loop 2-5, the *RAs* start by presenting proposals that give the highest utility and, then, proceed with proposals with lower utility. Agents have to content with presenting a proposal that is more likely to be accepted according to the feedback received from the *OA* (Definition 6.5) if they prefer to reach an agreement. These principles apply to all one-to-many negotiation that may occur simultaneously. The negotiation termination rule 3.1 and 3.2 of our protocol guarantees that this loop will end when only one proposal is presented in a round (transition from state 5 to 3) or when the negotiation deadline reaches a specific value (transition from state 2 to final).
4. In loop 1-3 in Figure 6.3, we are in the situation where the *RA* that is the initiator of the *inter-RA negotiation* receives a failure from each of the other *RAs*, meaning that they are not able to present a partial-solution. In this case, the initiator *RA* issues a new request using another candidate solution from its list. This is a finite list of candidate solutions ordered according to the initiator *RA's* preferences. If all *RAs* present a partial-solution, then the loop ends (transition from state 1 to 2). If there is no agreement the loop will end when the list is empty (all this according to the termination rule 3.5-6), going to the final state.
5. In loop 1-4 (and 1-5) in Figure 6.3, we have a very similar situation to the previous ones. The only difference is that instead of not receiving any partial solutions we receive only one partial-solution and there is no agreement because of that. The loop will end when all partial-solutions are received or when the list of candidate solutions is empty (termination rule 3.4-6).

Property 2: Given our negotiation model, a negotiation process using the protocol proposed in this chapter will converge to a better solution (if a solution exist) in each round having found the best solution through the process at the end of the negotiation (or, in the worst scenario, a solution equal to the first one found).

Proof: Our negotiation model includes a multi-attribute, multi-round with qualitative feedback process. To prove *property 2* we need to prove three features: (1) In each round, out of all proposals presented by each *RAs*, the round winner is the best one according to the *OA* preferences; (2) It is not possible for a proposal worst than the one which won the preceding round to win the subsequent round and (3) in each round the *OA* gives feedback to all proposals presented by the *RAs* guiding them to converge with the preferred *OA* values. We will use the *Definitions* in section 6.3 as well as the *Rules of Interaction* presented in section 6.6 to prove this property.

1. In each round the *OA* calculates the utility of each proposal according to Definition 6.12. The utilities are compared and the one that gives more utility to the *OA* is declared the round-winner according to Definition 6.13. This proves that the proposal that gives the highest utility to the *OA* is the round-winner.
2. In each round and after the round-winner is declared, the *OA* classifies each attribute of the losing proposals and provides feedback accordingly. The losing agents, if they wish, will present better proposals on the subsequent round and the winner agent will present the same proposal presented on the preceding round. Rule 4.3 (section 6.6) forbids the presentation of a previously submited proposal by a looser agent. Considering the above and since we have in subsequent rounds the proposal that won the preceding one, it is not possible to have as a round-winner a worst proposal (but an equal one is allowed).
3. Using the qualitative feedback provided by the *OA*, the *RAs* can improve their proposals for subsequent rounds, meaning that the subsequent proposals will more likely to be according to the *OA's* preferences.

6.10 Application Example

In the previous sections we have defined our proposed protocol in a more formal and abstract way. Our intention in this section is to show how we applied the concepts and abstractions to real-world application examples, helping to better understand how the protocol works. Since we proposed a protocol that is suitable for different kinds of environments we chose two distinct application examples:

The first one, *Disruption Management in Airline Operations Control* has a cooperative distributed problem solving environment with agents that are willing to cooperate but, at the same time, possess some degree of self-interest and rationality. A good motivation and contextualization is provided in papers (Castro & Oliveira, 2011) and (Castro & Oliveira, 2007).

The second one, *B2B E-Contracting* has a competitive environment with client agents that want to select the best suppliers of textile fabrics. The supplier agents are self-interested and compete with the other supplier agents. A good description and contextualization is given in (Joana Urbano, 2012).

6.10.1 Disruption Management in Airline Operations Control

We are going to show how we use a multi-agent system and our protocol to model the Disruption Management in the Airline Operations Control problem as presented in (Castro & Oliveira, 2007). Here, the authors propose a multi-agent system that represents an Airline Operations Control Center (AOCC), including the role of *Operations Manager* (the agent that authorizes the implementation of the solution found), *Aircraft*, *Crew* and *Passenger Recovery* agents (responsible for solving aircraft, crew and passenger problems, respectively). However, only the *Crew Recovery* part was implemented and, as such, the system is not able to find an integrated solution[1] to the problem.

Using our protocol we need to define first what a problem is and, consequently, what are the dimensions of the problem. In this case we have three dimensions: aircraft, crew and passenger.

[1] a solution that includes simultaneoulsy the aircraft, crew and passenger parts of the problem.

Following our definition of dimension (Definition 6.2) we have for the aircraft's, crew's and passenger's dimension, respectively:

$$d_1 = \langle airc, \{ad, ac\}, \{\aleph, \Re\}, \{VP_{ad}, VP_{ac}\}, \left\{\frac{ad}{max(ad)}, \frac{ac}{max(ac)}\right\}\rangle \tag{6.32}$$

$$d_2 = \langle crew, \{cd, cc\}, \{\aleph, \Re\}, \{VP_{cd}, VP_{cc}\}, \left\{\frac{cd}{max(cd)}, \frac{cc}{max(cc)}\right\}\rangle \tag{6.33}$$

$$d_3 = \langle pax, \{pt, pc\}, \{\aleph, \Re\}, \{VP_{pt}, VP_{pc}\}, \left\{\frac{cd}{max(cd)}, \frac{cc}{max(cc)}\right\}\rangle \tag{6.34}$$

Regarding the attributes of d_1, *ad* is the aircraft delay, *ac* the aircraft cost, VP_{ad} and VP_{ac} the preferred values for those attributes. *Max(ad)* and *max(ac)* are values defined at the execution time so that the scoring function has a value between [0,1]. For the other dimensions, *cd* is the crew delay, *cc* the crew cost, *pt* the passenger trip time and *pc* the passenger cost.

According to definition 6.1 the problem is represented by the n-tuple of the above dimensions, i.e, $P = \langle d_1, d_2, d_3\rangle$. A solution (Definition 6.4) is represented by a n-tuple of partial-solutions, one for each dimension. According to definition 6.3 a partial-solution for the aircraft dimension ps_1 is a n-tuple of attribute-values for the set A^1 of attributes of the dimension d_1, e.g., $ps_1 = \langle 25, 3000\rangle$ meaning that the aircraft delay *(ad)* would be 25 and the aircraft cost *(ac)* 3000. The partial-solutions ps_2 and ps_3, for the crew and passenger dimensions, are defined in a similar manner. So, for example, $S = \langle 25, 3000, 5, 2300, 30, 1000\rangle$ would be a possible solution to the problem.

It is important to point out that, in this AOCC domain, the value of a partial-solution represents and encodes a plan that should be applied to the environment. Continuing with the example above, regarding the partial-solution ps_1, the plan could be *"to use aircraft CSTNA to perform flight 100 instead of aircraft CSTNB that will perform flight 200"*. With the application of this plan to the environment we will have a delay of 25 minutes and a cost of 3000 monetary units. Regarding ps_2 the plan could be *"use crew member 323 to perform flight 110 instead of crew member 432 that is sick"*, implying a crew delay of 5 minutes and a crew cost of 2300 monetary units and, finally, for ps_3 the plan could be *"send disrupted pax John Doe in flight 565 from Lisbon to Porto and then flight 420 to Paris instead of flight 230 from Lisbon to Paris"*, implying that the trip time delay would be 10 minutes and the passenger cost 1000 monetary units.

For an *OA* to evaluate a solution *S* it is necessary to have a score function as indicated in the *Problem Analysis* component of the *OA* architecture. Basically, we just need to define the relative importance of each issue in each dimension by defining weights for each attribute and, finally, define the relative importance of each dimension in the problem. Using the attributes score functions as defined above we have the following (partial) scoring function:

$$V^p(S) = \alpha_1 \left(w_1 \left(\frac{ad}{max(ad)} \right) + w_2 \left(\frac{ac}{max(ac)} \right) \right) + \alpha_2 (\ldots) + \alpha_3 (\ldots) \tag{6.35}$$

with

$$\sum_{i=1}^{3} \alpha_i = 1 \; and \; \sum_{j=1}^{6} w_i = 1$$

The w_j represents the importance of each attribute in the dimension and α_i the importance of each dimension in the problem. Having the score function the *OA* utility for a received proposal is given according to Definition 6.12.

Likewise, a *feedback classification* formula (Definition 6.16) needs to be defined so that the *OA* is able to provide qualitative feedback to each attribute j of the proposals that lost the rounds, by calculating a delta for j in relation to a preferred value. It is possible to have different formulas for each dimension. For this application we decided to use the same formula for the three dimensions. Since we are in a cooperative environment we defined the formula as:

$$\Delta_j = Fc_o\left(O^t_{r\to o}[j]\right) = \frac{((O^t_{r\to o}[j]) - VP_j)}{VP_j}, \;\; with \;\; VP_j \neq 0 \tag{6.36}$$

We have defined $\alpha = 0.1$ and the set $QF = \{ok, high, low\}$. The classification (i.e. the qualitative feedback) of an attribute, is calculated according to (Definition 6.5):

- If $\Delta_j \geq -\alpha$ *and* $\leq \alpha \Rightarrow Class_j = \{ok\}$
- If $\Delta_j > \alpha \Rightarrow Class_j = \{high\}$
- If $\Delta_j < -\alpha \Rightarrow Class_j = \{low\}$

Taking as an example the aircraft cost attribute above (*ac*) with a value of 2000 and a preferred value of 2500, we have

$$\Delta_{ac} = \frac{(2000 - 2500)}{2500} = -0.2 \Rightarrow Class_{ac} = \{low\}$$

For a problem with these characteristics we need an organizer agent o_1 and, at least, three respondent agents r_1, r_2 and r_3, one for each dimension, i.e., aircraft, crew and passenger, respectively. It is perfectly possible to have more *RAs* with the same competence of any of the other agents. For the sake of simplicity in this example we are just going to have the minimum needed.

Our negotiation model (Definition 6.6) includes the set of organizer agents $O = \{o_1\}$ and the set of respondent agents $R = \{r_1, r_2, r_3\}$. The environment E is composed by several databases with the operational flight plan, aircraft and crew roster, passenger reservations and so on. The communication language *CL* used is FIPA-ACL and the domain language *DL* is a set of classes and objects that represent the concepts involved. The interaction protocol is the one we propose in this chapter.

From the point of view of the *OA* (o_1) the negotiation process is defined as $NegP^{o_1} = \langle o_1, \{r_1, r_2, r_3\}, \langle d_1, d_2, d_3\rangle, L^{o_1}\rangle$. From the point of view of the *RA* (r_1) the negotiation process is defined as $NegP^{r_1} = \langle r_1, \{r_2, r_3\}, o_1, \langle d_1, d_2, d_3\rangle, L^{r_1}, Q\rangle$. The r_2 and r_3 have a similar view of the negotiation process. The n-tuple Q represents some of the concepts necessary for the Q-Learning algorithm to be applied in this domain. These are related to the adaptability characteristics of the *RA*, i.e., the *State*, *Action* and the *Reward Function*.

A *State* (definition 6.17) for the *RA* (r_1) was represented as

$$ST^{r_1} = \langle evc, ret, Class_{ad}, Class_{ac}\rangle$$

$Class_{ad}$ and $Class_{ac}$ is the feedback classification (i.e, *ok*, *high* or *low*) received from the *OA* regarding attribute *ad* and *ac*, respectively. Regarding the attributes from the *problem domain*, *evc* represents the *cause of the event* (e.g. *aircraft malfunction*) and *ret* the affected resource type (e.g. *aircraft*). A similar state representation exists for the r_2 and r_3.

Regarding the *Action* representation (definition 6.18) for the *RA* (r_1) it was represented as

$$AT^{r_1} = \langle pda, at_{ad}, at_{ac}\rangle$$

$pda \in \{change, reroute, join, delay, acmi, cancel\}$, $at_{ad}, at_{ac} \in \{increase, decrease, keep\}$. pda is the *problem domain action* for this dimension. A similar representation exist for the r_2 and r_3. To conclude the definition of the Q-Learning concepts we have used the following reward function (Definition 6.19) for all *RAs*:

$$Rw_r = \begin{cases} m & \text{, if winner} \\ \frac{m}{2} - \sum_i^m pen_i & \text{, if looser.} \end{cases} \tag{6.37}$$

m is the number of attributes in a proposal and $0 \leq pen_i \leq 1$. Because we want a faster convergence, we have defined the penalization as follows:

- If $Class_i = \{high\} \Rightarrow pen_i = 0.8$
- If $Class_i = \{low\} \Rightarrow pen_i = 0.2$
- If $Class_i = \{ok\} \Rightarrow pen_i = 0.0$

Having defined the *State*, *Action* and *Reward Function* the agents are able to use the learning mechanism to learn (adapt) their bid strategies according to what we have described in Section 6.8.5.

To better understand how this works in this application example, we are going to show a specific example. Figure 6.6 shows how agent r_1 (the one with competence for the aircraft dimension) used the learning mechanism to prepare a new proposal in a specific round t of the negotiation. In the previous round *t-1* the agent was in state ST_1^{r1} as the result of presenting a proposal with a possible solution $S_{t-1} =< 30, 3000, 5, 1000, 5, 1500 >$ for which it received the feedback $F_{t-1} =< high, high, ... >$. To present a proposal in round t the following will happen:

- The action with highest probability according to the Boltzmann formula (Equation 6.30) was selected. In this specific case it was $AT_1^{r1} =< change, decrease, decrease >$ (please note that each *RA* only cares about the feedback received for the dimension on which it has competence. In this example $< high, high >$).
- Using action AT_1^{r1} as a reference the agent selected the partial-solution that is nearer to the action, i.e., one that decreases the values presented in the previous solution. In this example it was selected the partial-solution $ps_1 =< 25, 1500 >$.
- After engaging in an *Inter-RA* negotiation, agent r_1 presented a proposal with a possible solution $S_t =< 25, 1500, 10, 700, 10, 1600 >$ for which it received the feedback $F_t =< high, low, ... >$.
- The result of executing action AT_1^{r1} when it was at state ST_1^{r1} was the transition to state $ST_2^{r1} =< malfunction, aircraft, high, low, ... >$, obtaining the reward Rw_{r1} calculated according to Equation 6.37.
- The Q-Value in state ST_2^{r1}, i.e., $Q(ST_1^{r1}, AT_1^{r1})$ is updated according to Equation 6.31.

Since we used a respondent agent for each dimension we have defined the *Competence* (Definition 6.11) of each agent as follows (with $\alpha_1 = \alpha_2 = \alpha_3 = 0.5$):

- Aircraft dimension: $C_{r_1} = \left\langle airc, \{ad, ac\}, \{\aleph, \Re\}, \left\{ \alpha_1 \frac{ad}{max(ad)} + (1-\alpha_1) \frac{ac}{max(ac)} \right\} \right\rangle$
- Crew dimension: $C_{r_2} = \left\langle crew, \{cd, cc\}, \{\aleph, \Re\}, \left\{ \alpha_2 \frac{cd}{max(cd)} + (1-\alpha_2) \frac{cc}{max(cc)} \right\} \right\rangle$
- Passenger dimension: $C_{r_3} = \left\langle pax, \{pt, pc\}, \{\aleph, \Re\}, \left\{ \alpha_3 \frac{pt}{max(pt)} + (1-\alpha_3) \frac{pc}{max(pc)} \right\} \right\rangle$

Having the *Competence* of each respondent agent defined means they are able to participate in the main and Inter-RA negotiation according to the *Admission Rule* (**Rule 1** in Section 6.5).

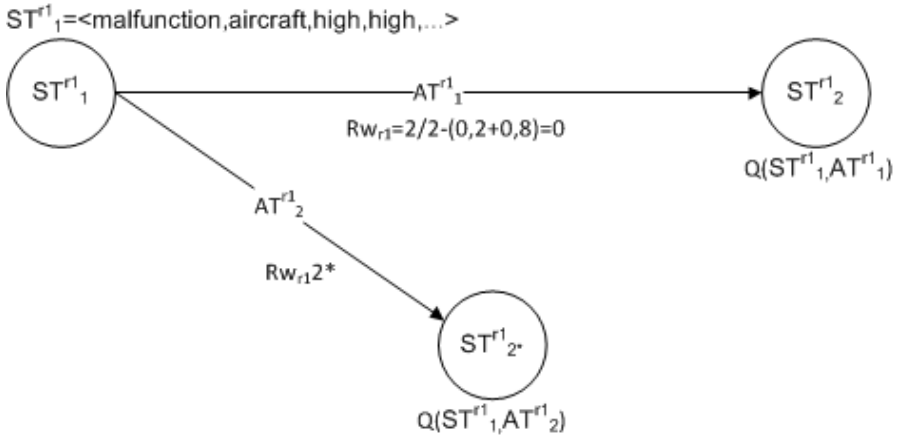


Fig. 6.6 Q-Learning in AOCC

Finally, we need to say something about the *Problem Solving* component (Section 6.8.5) included in the *RAs* we used. As we stated before, the partial-solutions represent and encode plans that, if applied to the environment, have the values indicated by the partial-solutions' attributes. To be able to get those plans, the respondent agents need to implement specific problem solving algorithms. In the case of r_1 (aircraft dimension) and r_2 (crew dimension) a Simulated Annealing algorithm (Kirkpatrick *et al.*, 1983) was used. In the case of r_3 (passenger dimension) the Dijkstra algorithm (Dijkstra, 1959) was used.

For the problem described in (Castro & Oliveira, 2007) we have developed MASDIMA a MAS whose description can be found in Appendix A. The MAS was developed in Java[2] and with the JADE[3] Framework. (Bellifemine *et al.*, 2004).

6.10.2 B2B E-Contracting

In this application example we are going to show how we use our protocol in a MAS to model the *B2B E-Contracting Textile Scenario* as presented in section 1.4 of (Joana Urbano, 2012). The authors have developed an agent-based framework for business-to-business e-contracting, which provides supporting services such as *automatic negotiation*, *contract monitoring* and *enforcement* and *computational trust*, although in this example our attention is focused only on the automated negotiation service.

In this scenario the textile client agents select the best suppliers of textile fabrics through a multi-round, multi-attribute negotiation. Every client has a business need which consists of a given component (fabric) and the associated preferential values for unit price, quantity and delivery time. A client agent announces its business need by issuing a call for proposal (CFP) to candidate suppliers and every supplier is able to provide any type of fabric. When a supplier agent receives a CFP concerning the provision of a given component, it generates a proposal based on its own preferential values for this specific component in terms of unit price, quantity and delivery times. This means that the client that issued the CFP has the expectation of receiving proposals from the candidate suppliers with diferent utility values.

[2] http://www.java.com/

[3] http://jade.tilab.com/

At the end of the negotiation phase, which involves the exchange of proposals and counter-proposals between the business partners (client and suppliers), the client agent finally selects the supplier whose proposal yields him the maximum utility.

Using our protocol we need to define what a problem is and, consequently, what are the dimensions of the problem. In this scenario there is only one dimension, since all agents are able (or have the full knowledge) to supply the required components. We are going to use only a component as an example, in this case, *voile*. For the other components presented in (Joana Urbano, 2012), similar representations are valid. So, we define the only dimension as:

$$d_1 = \langle voile, \{pr, qt, dl\}, \{I_{pr}, I_{qt}, I_{dl}\}, \{VP_{pr}, VP_{qt}, VP_{dl}\}, \{V_{pr}, V_{qt}, V_{dl}\}\rangle \tag{6.38}$$

with

$$I_{pr} = [min_{pr}, max_{pr}], \;\; I_{qt} = [min_{qt}, max_{qt}], \;\; I_{dl} = [min_{dl}, max_{dl}] \tag{6.39}$$

and

$$V_{pr} = \frac{(VP_{pr} - pr)}{(max_{pr} - min_{pr})}, \;\; V_{qt} = \frac{(VP_{qt} - qt)}{(max_{qt} - min_{qt})} \;\; V_{dl} = \frac{(VP_{dl} - dl)}{(max_{dl} - min_{dl})} \tag{6.40}$$

The attributes of d_1 are: *pr* for the price, *qt* for the quantity and *dl* for the delivery time. regarding the attribute domains, min_{pr} and max_{pr} are the minimum and maximum value of the price domain. For the other attributes the domain is defined in a similar way. VP_{pr}, VP_{qt} and VP_{dl} are the preferred values for the *price*, *quantity* and *delivery time* attributes.

The problem is represented by the n-tuple of the above dimension, i.e., $P = \langle d_1 \rangle$ and a solution is represented by the n-tuple of the partial-solution for this dimension, e.g., $ps_1 = \langle 6, 270000, 20 \rangle$ meaning that the price *(pr)* would be 6 (monetary units), the quantity *qt* 270000 (meters) and the delivery time *(dl)* 20 days. So, for example, $S = \langle 6, 270000, 20 \rangle$ would be a possible solution to the problem.

For an *OA* (a *client* in the terminology used in (Joana Urbano, 2012)) to evaluate a solution S it is necessary to have a score function. Using the attributes' score functions as defined above we have the following score function:

$$V^p(S) = \alpha_1 \left(w_1 \left(V_{pr} \right) + w_2 \left(V_{qt} \right) + w_3 \left(V_{dl} \right) \right) \tag{6.41}$$

with

$$\sum_{i=1}^{1} \alpha_i = 1 \; and \; \sum_{j=1}^{3} w_i = 1$$

The w_j represents the importance of each attribute in the dimension and α_i the importance of each dimension in the problem. In this specific scenario we only have one dimension. Having the score function the *OA* utility for a received proposal is given according to Definition 6.12.

The *feedback classification* formula (Definition 6.16) needs to be defined so that the *OA* is able to provide qualitative feedback to each attribute j of the proposals that lost the round. In this competitive scenario the authors of (Joana Urbano, 2012) defined the formula as follows:

$$\Delta_j = Fc_o \left(O^t_{r \to o}[j] \right) = \frac{(VP_j - (O^t_{r \to o}[j]))}{(max_j - min_j)} \tag{6.42}$$

To be able to provide qualitative feedback in this scenario, the authors calculate deltas for each attribute of the best proposal in the round and for the corresponding attribute of the other proposals,

e.g., let us call Δ_j^b to the delta of attribute j for the best proposal and Δ_j^k to the delta of attribute j for one of the other proposals. The set QF was defined as $\{excellent, sufficient, bad, verybad\}$ and the classification of an attribute, is calculated according to the following:

$$\Delta = \frac{\left(\Delta_j^k - \Delta_j^b\right)}{\Delta_j^b} \tag{6.43}$$

- If $\Delta < 0.005 \Rightarrow Class_j = \{excellent\}$
- If $\Delta < 0.05 \Rightarrow Class_j = \{sufficient\}$
- If $\Delta < 1.5 \Rightarrow Class_j = \{bad\}$
- If $\Delta > 1.5 \Rightarrow Class_j = \{verybad\}$

This qualitative feedback is sent to all respondent agents except to the one that presented the best proposal on each round.

Taking as an example the price attribute above (pr) and assuming that the best proposal on the round had a value of 3.0 for this attribute and one of the other proposals a value of 3.1 and that the preferred value is 5.0 and the domain is $[1,10]$, we have

$$\Delta_{pr}^b = \frac{(5.0-3.0)}{(10.0-1.0)} = 0.222$$

$$\Delta_{pr}^k = \frac{(5.0-3.1)}{(10.0-1.0)} = 0.211$$

$$\Delta = \frac{(0.211-0.222)}{0.222} = -0.050 \Rightarrow Class_{pr} = \{excellent\}$$

In the scenario presented in (Joana Urbano, 2012) the authors include two more components (*cotton* and *chiffon*) besides the one we have used as an example above (*voile*). Additionally, they used 10 clients (each one interested in one of the components) and 20 suppliers that are able to trade any of the components (meaning that each supplier has three competences). For a scenario like this our negotiation model (Definition 6.6) includes the set of organizer agents (the *clients*) $O = \{o_1, o_2, \ldots, o_{10}\}$ and the set of respondent agents (the *suppliers*) $R = \{r_1, r_2, \ldots, r_{20}\}$. The environment E is composed by the attribute domain values of each agent and the communication language CL used is FIPA-ACL. The domain language DL is a set of classes and objects that represent the concepts involved. The interaction protocol is the one we propose in this chapter.

Continuing to use the *voile* component as an example, from the point of view of one of the clients interested in this component (e.g., the *OA* o_1) the negotiation process is defined as $NegP^{o_1} = \langle o_1, \{r_1, r_2, \ldots, r_{20}\}, \langle d_1 \rangle, L^{o_1} \rangle$.

From the point of view of one of the suppliers (e.g., the *RA* r_1) the negotiation process is defined as $NegP^{r_1} = \langle r_1, \{\}, o_1, \langle d_1 \rangle, L^{r_1}, Q \rangle$. The other *RAs* have a similar view of the negotiation process. The n-tuple Q represents some of the concepts necessary for the Q-Learning algorithm to be applied to this domain. These are related to the adaptability characteristics of the *RA*, i.e., the *State*, *Action* and the *Reward Function*.

A *State* (definition 6.17) for each of the *RAs* was represented as

$$ST^r = \left\langle Class_{pr}, Class_{qt}, Class_{dl} \right\rangle$$

$Class_{pr}$ is the feedback classification (i.e, *excellent*, *sufficient*, *bad* or *verybad*) received from the *OA* regarding attribute *pr* and the others have the same meaning.

Regarding the *Action* representation (definition 6.18) for the *RA* (r_1) it was represented as

$$AT^{r_1} = \langle at_{pr}, at_{qt}, at_{dl} \rangle$$

$at_{pr}, at_{qt}, at_{dl} \in \{incFew, incMedium, incVery, decFew, decMedium, decVery, maintain\}$. A similar representation exists for the other *RAs*. To conclude the definition of the Q-Learning concepts we have used the following reward function (Definition 6.19) for all *RAs*, considering that a proposal has three attributes:

$$Rw_r = \begin{cases} 3 & \text{, if winner} \\ \frac{3}{2} - \sum_i^3 pen_i & \text{, if looser.} \end{cases} \tag{6.44}$$

The penalization was defined according to ($0 \leq pen \leq 1$):

- If $Class_i = \{verybad\} \Rightarrow pen_i = 0.9$
- If $Class_i = \{bad\} \Rightarrow pen_i = 0.6$
- If $Class_i = \{sufficient\} \Rightarrow pen_i = 0.2$
- If $Class_i = \{excellent\} \Rightarrow pen_i = 0.0$

Having defined the *State*, *Action* and *Reward Function* the agents are able to use the learning mechanism to learn (adapt) their bid strategies.

In this example we have one dimension. So, every *RA* has the same competence that we have defined as follows (with $\alpha = 1$):

$$C_{ra} = \langle voile, \{pr, qt, dl\}, \{I_{pr}, I_{qt}, I_{dl}\}, \{(1/\alpha)V_{pr} + (1/\alpha)V_{qt} + (1/\alpha)V_{dl}\} \rangle$$

See Equation 6.39 for the definition of I_{pr}, I_{qt} and I_{dl}, and Equation 6.40 for the definition of V_{pr}, V_{qt} and V_{dl}. It is important to point out that it is possible for the same *RA* to participate in another parallel negotiation with another *OA* and with a different competence.

In this particular scenario, the *RAs* are all suppliers, i.e., agents of the same type. However, they have different preferred values as well as different range of possible values for each attribute under negotiation. Additionally, some of the *RAs* have a *kind of handicap*. For example, some of them might have an *handicap* in providing high quantities or low delivery times. This makes the *Problem Solving* component of each *RA* in this scenario, very simple (specially when compared with the application example presented in section 6.10.1): when looking for candidate solutions to present a new proposal, any combination of values within the range of possible values is valid. To present a new proposal, considering the feedback received, the *RAs* change the value of the attribute (up or down), according to the following:

- If $Class_i = \{verybad\}$ the value is changed 10%.
- If $Class_i = \{bad\}$ the value is changed 5%.
- If $Class_i = \{sufficient\}$ the value is changed 0.5%.
- If $Class_i = \{excellent\}$ the value is not changed.

6.11 Advanced Features

The purpose of this section is to identify features that, although not conceptualized in this chapter or used in the experimental section of this thesis, have been used in other studies conducted in parallel with our work. The main groups of features are:

- *Open Environments*: The GQN protocol has been used in closed environments. We are working in improvements that will make the protocol more suitable to be used in open environments, e.g., by including an Ontology Service component, similar to the one of (Malucelli *et al.*, 2006), which would allow using agents made by different designers to make the match between problem dimensions and agent competences in a more dynamic way. This component would also allow to define more dynamically the negotiation object, i.e., the set of attributes to be used in the negotiation.
- *Organizer Agent Learning*: Our protocol already includes adaptable strategies through the inclusion of Q-Learning during proposal formulation on the respondent agents (RA). We are working in including learning in the organizer agents (OA). We believe that the *OA* could "learn with the past" and use this information before starting a new negotiation. Additionally, since we plan to include *Human-in-the-loop* features (see Chapter 7) the *OA* could use the feedback information received from the human to adapt its strategy not only before starting a new negotiation but, also, during the negotiation.
- *Argumentation*: The GQN protocol already includes qualitative feedback. However we believe that the inclusion of arguments in our model will support more negotiation scenarios, making it more suitable to be used in more real world applications. We are changing our model to include *argument interpretation*, *argument generation* and *argument selection*.
- *Normative Environment and Trust*: Finally, through the integration of our protocol in the ANTE framework (Oliveira, 2012; Lopes Cardoso *et al.*, 2013). In doing so, we may take advantage of a normative environment and computational trust, two characteristics that, in our opinion, will enrich the model.

6.12 Chapter Summary

In this chapter we have presented the Generic Q-Negotiation (GQN) protocol. A protocol that will help us to draw conclusions about the *second hypothesis* as formulated in Section 1.3.2. We call it *Generic* because it can be used in very different and heterogeneous environments. The *Q-Negotiation* part of the name comes from the fact that it inherits the characteristics of the Q-Negotiation Protocol (Rocha & Oliveira, 1999) an adaptive protocol that includes the Q-Learning algorithm. GQN, the proposed protocol has four characteristics that, in our opinion, allow it to be applied to more application domains (making it more generic):

1. it supports agents with heterogeneous behavior (covers more types of environments).
2. it supports agents with full and/or partial knowledge (useful for distributed environments).
3. it includes agents with adaptive behavior (useful for dynamic environments).
4. it supports multidimensional negotiations with interdependent dimensions.

GQN is an adaptive protocol for multi-attribute negotiation with several rounds and qualitative feedback, with interdependent dimensions, supporting agents with full or partial knowledge of

two types, i.e., organizer and respondent agents. When needed, the *RAs* negotiate with each other to "build compatible solutions to their interdependent sub-problems". We believe that this *Inter-RA Negotiation* addresses the first problem reported by Durfee et al. The agents are also able to learn (adapt) their strategies during bid formulation, due to the inclusion of a Q-Learning algorithm. We believe that this characteristic adresses the "changing of behavior" problem also reported by Durfee et al.

Additionally, we show how the GQN could be used to model two very different application domains and present some advanced features that could be part of it in the future. In the next chapter we will present another of the main contributions of our work: *A New Concept for Disruption Management in Airline Operations Control*.

Acknowledgements We wish to thank Ana Paula Rocha for the enlightening discussions about the GQ-Negotiation protocol as well as for the work we have done together. We are also grateful to Henrique Lopes Cardoso for the help in reviewing this chapter and to Professor Cristina Câmara for the help in reviewing some of the definitions presented here.

Chapter 7
A New Approach for Disruption Management in AOCC

Abstract In Chapter 4 we introduced the Airline Scheduling Problem and the Airline Operations Control Problem. We have described the AOCC organization and its roles as well as the typical problems that appear during the execution of the operational plan. The disruption management process used by airlines was presented as well as the main costs involved in generating and evaluating the solutions. In this chapter[1] we present our new approach to disruption management in the airline domain, including how we represent the AOCC using a Multi-Agent System (MAS), a computational organization of intelligent agents. Besides providing new or updated processes and mechanisms for the AOCC we also aim to focus on the AOCC as an entity, trying to change the way AOCCs are setup and organized and not only in providing DSS tools. Our main focus in this chapter is on the conceptual aspects of our approach. However, although we tried to be as concise and objective as possible, sometimes we had to give more detail so that the reader can better understand the scope of our proposal. The work presented in this chapter will help us to draw conclusions about the *first, third* and *fourth hypotheses* as formulated in Section 1.3.2.

7.1 Introduction

In this chapter we propose a new approach for disruption management in the airline domain. Our vision for the future AOCC is presented in the following paragraph:

> We see the AOCC as an *Autonomous* and *Automatic Entity* that consumes information from several sources, monitors that information, solves the problems and makes decisions according to the local and global objectives. This entity updates the operational plan according to the solutions found and the decisions taken and learns with the past to avoid similar problems in the future.

We know that, for now, this is an utopic vision. However, we think that it is possible to improve substantially the current organization, approach and tools in the AOCC. We see the AOCC as an organization with local goals (for example, minimizing the costs with aircraft, crew and/or passengers when solving a specific disruption) but also with global goals like minimizing delays and costs in a given period of time. The objective is to make the AOCC more efficient, faster when solving disruptions and with better global decisions and performance. We believe that human experts should be managers and not controllers. In our opinion, repetitive or frequent tasks are better performed by software agents and tasks with a high degree of uncertainty are performed better by humans.

From the analysis we have done on current AOCC (Chapter 4) we think that a distributed, scalable and cooperative approach can greatly improve the problems faced by such an organization. By these characteristics we mean:

[1] Parts of the material found in this chapter was published in several papers and book chapters, specifically (Castro & Oliveira, 2009, 2010, 2011).

- *Functional Distribution*: To distribute the roles and functions that exist on the AOCC. Nowadays, the airline companies already distribute the roles and functions by several persons and teams. We think that we can do the same and maybe better, using intelligent software agents.
- *Spatial Distribution*: This is related to the location of data. If all data is in the same place, then it is not distributed. If it is in different places, e.g., in different databases, or in the same place but clearly divided, e.g., in the same database but in different partitions, then it is distributed. In our opinion the AOCC can also benefit from this.
- *Physical Distribution*: Having the roles or functions assigned to software agents it is possible and useful to distribute them in different machines. This way, there is more processor power and resources that can be used by each of the software agents.
- *Scalability*: The possibility to grow according to the functional and/or spatial needs as well as according to any other domain related requirement.
- *Cooperative*: Although the AOCC has different teams responsible for different functions and, as such, with different goals, the AOCC by itself also has overall goals. An approach that could guarantee that the final results contribute more for the common good, i.e., to achieve the AOCC common goals and, at the same time, without forgetting the local ones, would be a strong asset.

Additionally, in Section 4.8 we have also identified the following requirements:

- The different views that exist in the AOCC, i.e., *aircraft*, *crew* and *passenger*, should have the opportunity to be considered at the same level of importance, when looking for solutions to the problems.
- The local preferences of each team in the AOCC should be considered.
- The solutions should be found in real or almost real-time.
- The dynamics of the environment should be taken into consideration, since the existing information can unexpectedly change.
- The information and time available to get a solution may not be complete or limited.

With these ideas in mind and considering the above requirements, we propose to represent the AOCC as an organization of agents, a multi-agent system (MAS), where the roles that correspond to the most frequent tasks that could benefit from a cooperative approach are performed by intelligent agents. The human experts, represented by agents are able to interact with them, are part of this AOCC-MAS supervising the system and making the final decision from the solutions proposed by the AOCC-MAS. We believe that what we propose in this chapter is a large step *Towards an Advanced Autonomous Integrated Airline Operational Control Center*.

The rest of this chapter is organized as follows. In Section 7.2 we present the characteristics of the agent and multi-agent system paradigm that make us adopt it. In Section 7.3 we show the general and main ideas behind the *autonomous and integrated* AOCC we propose. Section 7.4 is about the MAS architecture, either with single or multiple instances and in Section 7.5 we presented the operational costs involved. In Section 7.6 we show the decision mechanisms including the adaptive characteristics we used in the MAS and in Section 7.7 the specification of problem solving algorithms used by some of the agents. In Section 7.8 we summarize the advanced features that, although not implemented, have been studied or considered relevant to be included in the future. Finally, we end with a Chapter Summary in Section 7.9.

7.2 Why an Agent and Multi-Agent System Paradigm?

Before presenting the architecture of our multi-agent system, it is important to point out the characteristics of this paradigm, according to (Wooldridge, 2009a) and (Elamy, 2005), that make us adopt it to model this problem:

- *Autonomy*: MAS models problems in terms of autonomous interacting components, which are a more natural way of representing task allocation, team planning, and user preferences, among others. In Figure 7.2 the *PaxManager*, *AircraftManager* and *CrewManager* agents (among others) are agents that can choose to respond or not to the requests according to their own objectives.
- *Agents are a Natural Metaphor*: The AOCC is naturally modeled as a society of agents cooperating with each other to solve such a complex problem. That is also sustained by the analysis and design that was performed before implementing the MAS we have developed.
- *Reactivity*: Agents are able to perceive and react to the changes in their environment. The *Monitor* agent in Figure 7.2 is an example of such an agent.
- *Distribution of resources*: With a MAS we can distribute the computational resources and capabilities across a network of interconnected agents avoiding problems associated with centralized systems. Airline companies of some dimension have, for example, different operational bases. We can use a MAS for each operational base, taking advantage of this important characteristic. Due to the *social-awareness*[2] characteristics of some of our agents (for example, *Monitoring* agent in Figure 7.2) they are able to distribute their tasks among other agents with similar behavior. However, there are other possibilities for distribution as we stated in the previous section.
- *Modularity and Scalability*: A MAS is extensible, scalable, robust, maintainable, flexible and promotes reuse. These characteristics are very important in systems of this size and complexity. Our MAS is able to *scale*, for example, in terms of supporting more operational bases as well as in supporting different algorithms to solve specific problems.
- *Concurrency/Parallelism*: Agents are capable of reasoning and performing tasks in parallel. This provides flexibility and speeds up computation. The *Specialist* agents Figure 7.2 are examples of concurrent agents. Additionally and according to (Stone & Veloso, 2000) "if control and responsibilities are sufficiently shared among agents, the system can tolerate failures by one or more agents". Our MAS can be totally or partially replicated in different computers. If one or more agents fail, the global objective is not affected.
- *Legacy Systems*: The AOCC needs information that exists in obsolete but functional systems. We can wrap the legacy components on an agent layer, enabling them to interact with other software components.

7.3 Towards an Advanced and Autonomous Integrated AOCC

Considering the characteristics and advantages of the MAS paradigm presented on the previous section and looking at the current roles in the AOCC (Figure 4.3 in Section 4.3), we see that some of them correspond to very repetitive tasks. For example, the aircraft controller (a member of the *aircraft team*) is constantly checking the computer system (including, e-mail, *datalink* system,

[2] an awareness of other agents in the environment and knowledge about their behavior (Kazakov & Kudenko, 2001).

telex, etc.) to see if there is any problem that might affect the departure or arrival of a flight. A similar routine regarding monitoring crew members is performed by the crew controller (a member of the *crew team*) and by the members of the *passenger team*.

When a problem is detected, the process of solving it is also very repetitive. For example, if a flight is delayed, the *aircraft controller* follows the *General Process of Aircraft Resolution* as presented in Section 4.6 which includes the following actions (see also Table 4.4):

1. Use an aircraft from a later flight (exchange aircraft).
2. Reroute the flight.
3. Join flights (use one aircraft to also perform the disrupted flight).
4. Lease an aircraft and crew from another company (ACMI).
5. Delay the flight.
6. Cancel the flight.

Additionally and as presented in Table 4.5 of Section 4.6 they select these actions according a probabilistic relation with the type of event that caused the problem.

A similar approach, although with different actions and probabilities, is followed by the members of the *Crew team* and the *Pax team* as it is possible to see in Tables 4.6 and 4.7, respectively. Even the role and function of the *human supervisor* is partially repetitive. Basically, he is called to make decisions when there is no consensus among the several teams on how to solve a specific problem.

Taking into consideration the above as well as the characteristics of the agent and multi-agent paradigm as presented in the previous section, we propose to represent the AOCC by a multi-agent system, replacing the monitoring, aircraft team, crew team, passenger team and part of the supervisor, maintenance services and flight dispatch role, by intelligent agents as represented in Figure 7.1.

In this new approach, the aircraft team will be replaced by a sub-organization of agents (represented as *Aircraft Manager*). The same will happen to the crew team (represented as *Crew Manager*) and to the passenger team (represented as *Passenger Manager*). Regarding the *passenger services*, we propose to replace by software agents the task of finding the best solutions to the problems with passengers (usually a plan of alternative flights to each disrupted passenger) and keep the other tasks to be performed at the airports by human operators. The *supervisor* interacts with the software agents through an interface agent and will see its role improved since it will be able not only to make the final decision regarding a solution proposed by the MAS but, also, to provide feedback regarding that solution. This way, it will be able to teach the system to react better not only in similar problems that might happen in the future but, also, in the specific problem that is being solved. This *feature* is called *Human-in-the-Loop*. The *maintenance services* and *flight dispatch* will also interact with the software agents through an interface agent. This will facilitate the input of data related to aircraft events as well as the input and output of data related to flight preparation, respectively.

Finally, in Figure 7.1 we have also included six features (in gray and numbered from 1 to 6) that, although not implemented in the MAS and, as such, not included in the experimentation we have performed, are important and will contribute to the goal of having an *Advanced and Autonomous Integrated AOCC*. A brief summary of these features follows:

1. *Maintenance Personal Tablet PCs*: The communication between the *Maintenance Services* team in the AOCC and the maintenance personnel at the airport can greatly be improved through the

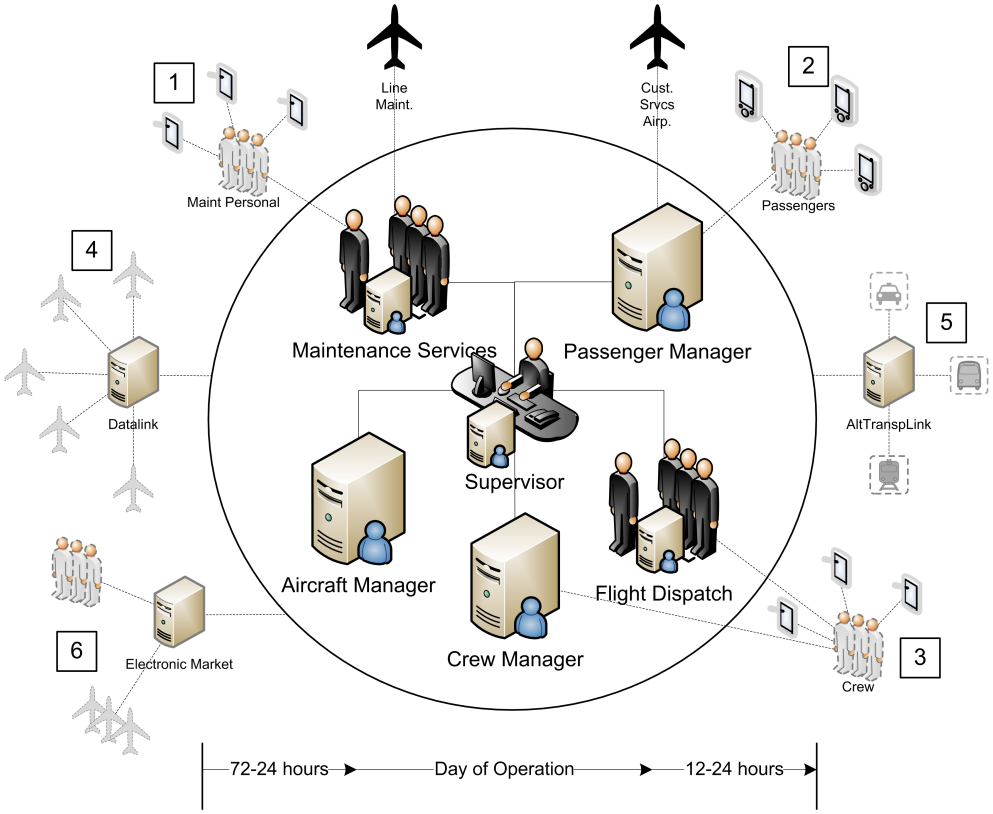


Fig. 7.1 New Concept for an Integrated Airline Control Center

use of tablet pc's. It will be quicker and with less error to report events to the AOCC and receive any decisions.

2. *Passengers Smartphones*: According to a SITA[3] report (Sita, 2010) there are five things that smartphones will change by the year 2020. To receive alerts regarding disruptions, including new ETD and itineraries is one of them.
3. *Crew members Tablet PCs*: Nowadays the use of personal computers by the pilots to perform some of their operational tasks is already a reality. The Electronic Flight Bag (EFB) is such an example. A better and wider integration can be achieved and the same concept can be extended to cabin crew members.
4. *Aircraft Datalink Connection*: ACARS communication is already a reality in the Air Transport Industry. We think that it is possible to improve the work of the AOCC through a better use of this tool, specially regarding a proactive action towards avoiding flight arrival delays.
5. *Alternative Transportation Datalink*: This is one of the most advanced features we propose for the future. Usually, the solution for a disrupted flight from the point of view of the passenger, is to find an alternate flight itinerary. For the majority of flight disruptions, this is enough. In a matter of few hours it is possible to find alternate flights. However, as it was proved by the 2010 Icelandic volcanic ash that caused flight disruptions that took days to be solved, under certain conditions, to have alternate itineraries performed by different means of transportation (e.g., bus, train) might be an acceptable solution. Using a MAS is relatively simple and straight forward to contemplate these means of transportation as possible solutions to the problem.

[3] Provider of global information and telecommunication solutions for the air transport industry (http://www.sita.aero).

6. *Aircraft and Crew Electronic Market*: This is the last advanced feature we propose. The AOCC could use a broker to an *Electronic Market* of aircraft and crew members as a possible solution to a flight disruption. Nowadays, it is common for the airline companies to lease an aircraft and crew in some situations (called ACMI). In a 2006 paper (Malucelli *et al.*, 2006) we present some preliminary work regarding such a market.

7.4 MASDIMA Architecture

Before presenting the architecture of our system it is important to point out that we have followed an agent-oriented methodology to analyze and design the MASDIMA. The methodology called *PORTO*, described in Chapter 5, was used to perform these tasks. The main deliverables of the analysis and design we have done, are the *Agent Model* and the *Service Model*. These models allow the implementation of the MAS in the language of choice by the system designer or developer, since they are *programming language independent*.

In this section we are going to present the architecture of the system, based on the agent and service model and using a *free-form graphical* notation to avoid cluttering the drawing space with unnecessary details for the level of representation we want for now. We start by presenting the *single-instance agents* and, then, the *multiple-instance agents*.

7.4.1 Single-Instance Agents Architecture

Figure 7.2 shows the *Single-Instance Agent* architecture of our multi-agent system. By *Single-Instance Agent* we mean an architecture where we have only an instance of each agent. A *Multiple-Instance Agent* architecture is presented in Section 7.4.2. For simplicity reasons we did not include the agents that are specific to the JADE framework, such as, *AMS* or *DF*. Nevertheless and since the MAS uses these agents, we present in Table 7.1 a description as well as some of the services provided by those agents (for more information regarding the JADE runtime framework please consult (Bellifemine *et al.*, 2004)).

In Figure 7.2 the solid boxes represent agents that we have implemented in our prototype and, from those, the ones with round corners represent user interface agents. The solid lines represent interactions between agents and the dashed lines represent actions in the environment. The cloud represents negotiation at the managers' level. The ellipse represents the environment with the information data sources identified in Section 4.4. The dashed gray boxes represent agents that were not implemented on our prototype. However, we chose to put them here because they help to understand how we would implement the advanced features we have referred in Figure 7.1. More information about these agents appears in Section 7.8.

7.4.1.1 Agent Roles

Each agent performs one or more roles in the AOCC. The *Monitor* agent looks for events on the *operational plan*[4] that may trigger any aircraft/flight, passenger and/or crew problem. This agent

[4] By *operational plan* we mean the flight/aircraft schedule as well as the related crew and passenger information.

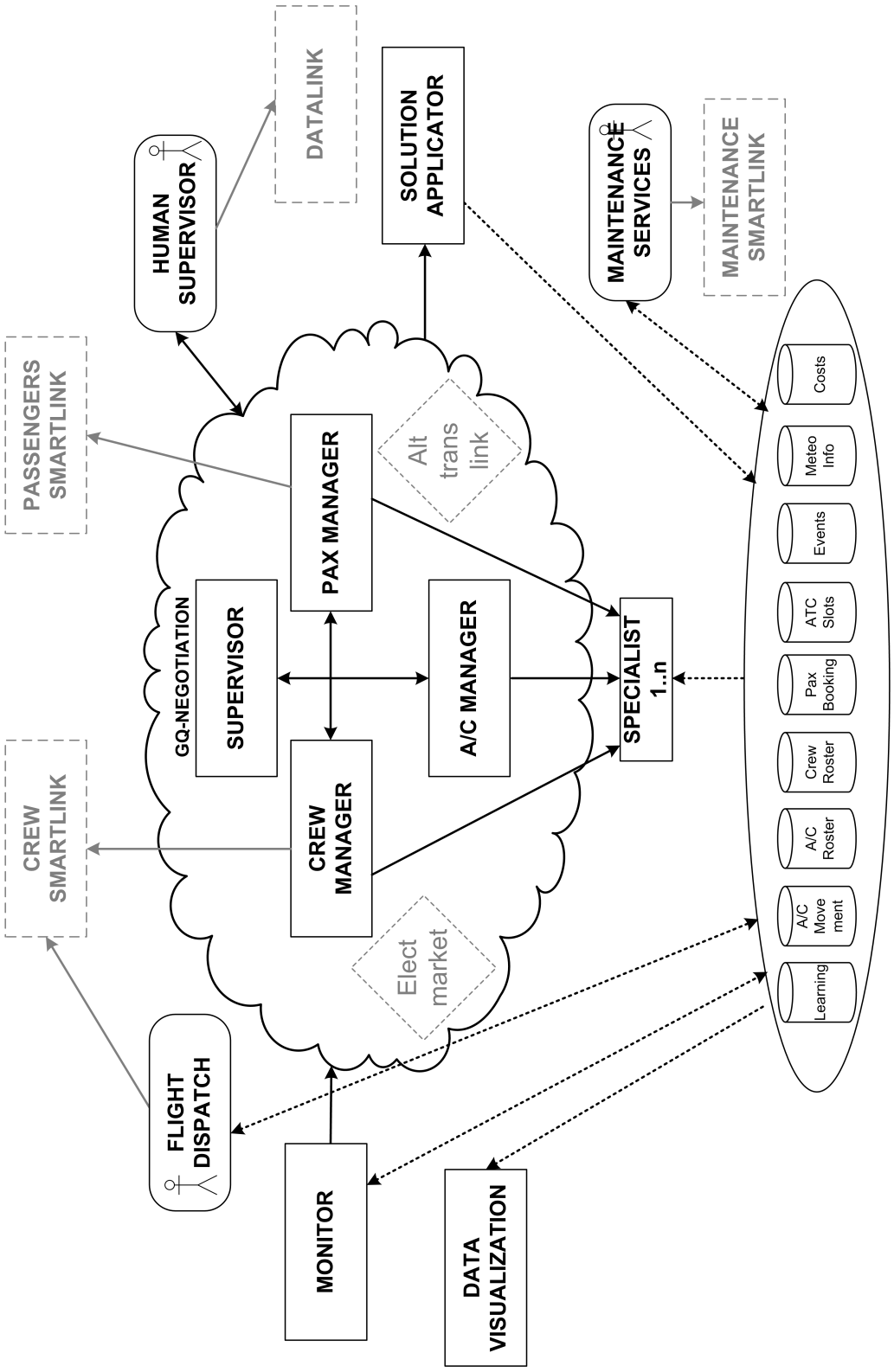


Fig. 7.2 Multi-Agent System Architecture

Table 7.1 JADE Specific Agents

Agent	Description	Methods
AMS	*Agent Management System* supervises the use and access to the platform and keeps a directory of the agents and its state. All agents have to register through the AMS.	⇒ *register* called automatically to register an agent; *deregister* called automatically to unregister an agent.
DF	*Directory Facilitator* it is a *Yellow Pages* service. Allows the agents to register their services, search for agents that provided a specific service and subscribe notifications every time a given service is published.	⇒ *DFRegisterAgent* is the method that allows an agent to register a specific application service; *DFSubscribeAgent* allows an agent to be notified each time a given service is published in the yellow pages catalog; *DFSearchAgent* allows an agent to search for services provided by other agents and advertised in the yellow pages catalog.

has *social-awareness* characteristics in the sense that it is able to recognize and interact with other agents with the same role, sharing the tasks. This agent, like others in our system, is *autonomous* because it is able to consider an event as a problem only when specific conditions or characteristics are present.

The *Crew Manager*, *A/C Manager* and *Pax Manager* agents are responsible for crew, aircraft/flight and passenger problems, respectively. They manage a team of expert agents with the role of finding solutions for the problems in their area of expertise. The expert or specialist agents implement different heterogeneous problem solving algorithms and are able to run in parallel. The managers are *autonomous* because they can choose to respond or not to requests related to their area of expertise. The *Supervisor* agent automates part of the human supervisor role in Figure 7.1. This agent together with the three managers engage in an automated negotiation using the GQN protocol (see Chapter 6) with the goal of getting the best integrated solution[5] for a specific problem. The solution(s) that come up from this negotiation process will be presented to the *Human Supervisor* for approval and feedback.

7.4.1.2 Interface Agents

The agent *Human Supervisor* is an user interface agent and, as such, interacts with the human supervisor of the AOCC. As stated before, the solutions selected by the *Supervisor* are presented to the *Human Supervisor*. It includes solution details (and the rationale behind the solution) to help the human decide and are ranked according to the criteria of the airline company (i.e., according to the supervisor utility 7.4). The *Human Supervisor* can approve or not the solution and provide feedback (more information about this part will be given in Section 7.6). After getting approval the *Supervisor* agent requests *Solution Applicator* agent to apply the solution to the environment. The *Solution Applicator* agent has a very important role because it is responsible for checking if the state of the environment regarding the operational plan (aircraft and crew roster, for example) keeps the initial conditions that lead to the solution found and approved. Otherwise, if the solution

[5] the one that takes into consideration the three dimensions of the problem simultaneously.

is applied without guaranteeing this condition we could get an inconsistent operational plan. In the case of not being possible to apply the solution the *Solution Applicator* agent informs the *Supervisor* and a new negotiation will start. It is important to point out that the *Solution Applicator* agent has some flexibility in checking the conditions and applying the solution. It is not a *strict enforcement* of the original conditions otherwise we could end up in a loop between the *Solution Applicator* and *Supervisor* agent.

The agents *Flight Dispatch* and *Maintenance Services* are also user interface agents. The former allows the flight dispatch team of the AOC to enter relevant information about each flight. For example, flight plans, ATC slots, NOTAMS, etc. It also has the possibility to interact with the (future) *Crew Smartlink* agent to provide information to the crew members directly to their tablet pc and/or smartphones. The latter allows the maintenance services team of the AOC to enter relevant information about the aircraft. For example, any unexpected maintenance check, aircraft malfunction event, etc. It also has the possibility to interact with the (future) *Maintenance Smartlink* agent to provide and receive information directly from the line maintenance personal at the airports.

The *Data Visualization* agent supports the visualization of the information (flight movements, delays, events, solutions to the problems, etc.) showing what is happening at the AOCC. Figure 7.3 shows a partial GUI updated by the *Data Visualization* agent. More information about the implemented MAS, called MASDIMA (Multi-Agent System for Disruption Management) can be found in Appendix A.

Finally, in figure 7.2, the cylinders represent the *Data Sources* available on the environment for the agents to observe and act upon. All the necessary information is included in the data sources. Table 4.1 in Section 4.4 gives details about these data sources. Additional information to support some characteristics of the MAS like *learning* is also included on the data sources. The learning abilities are part of the *Supervisor* and *Manager* agent's strategy and details are presented in Section 7.6.

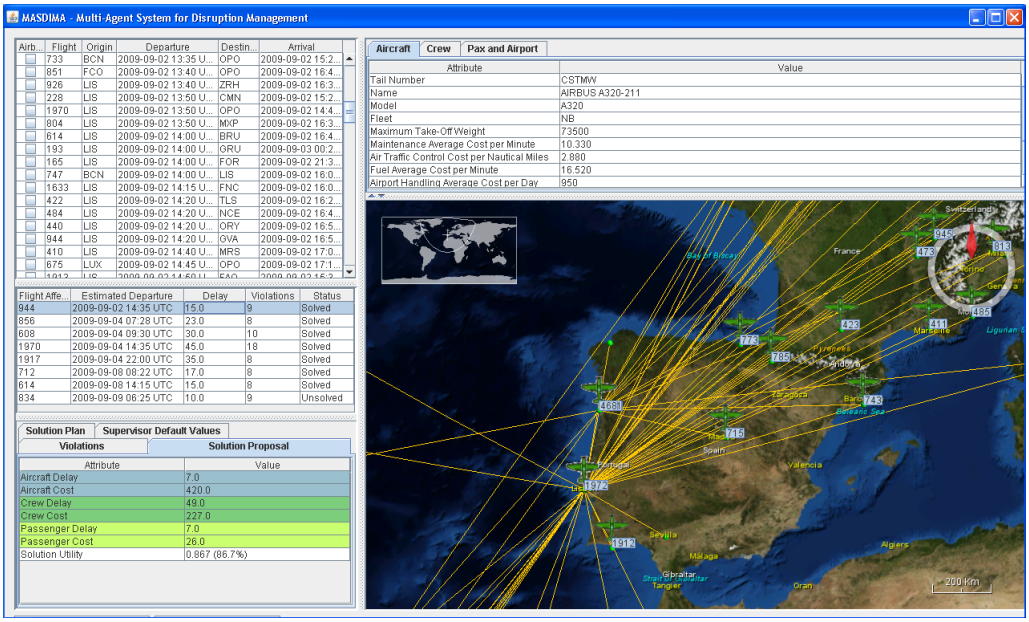


Fig. 7.3 Multi-Agent System GUI Example

7.4.2 Multiple-Instance Agents Architecture

On the previous section we have presented a multi-agent system architecture with only a *single-instance* of each agent and with each agent deployed in the same machine. By single-instance we mean *a single copy of a running agent program*. As we stated previously, it is possible and for some airline companies desirable, to have a more distributed architecture, either *Functional*, *Spatial* and/or *Physical*. Several combinations are possible, including to have a single-instance agent architecture deployed in a distributed way (some agents hosted in one machine and the others on another) and a multiple-instance agent architecture deployed in a non-distributed way (all agents in the same machine).

Here, we are going to present a multiple-instance agent architecture deployed in a distributed way that could be used by an airline with two operational bases (i.e., each operational base with a specific number of aircraft and crew members based there and a specific number of flights) being one of the operational bases the main *hub*.

Figure 7.4 presents the proposed *Multiple-Instance Agent* architecture. The *Functional Distribution* is obtained by distributing different instances of the agents with the same role by two operational bases. All the agents on operational base B are different instances of agents with the same role on operational base A. For example, the *MT:B* (Monitor) agent is another instance of the *MT:A* agent.

The *Spatial Distribution* is obtained by partition of the data: one part regarding the operational base B data and another regarding the operational base A data. The agents on the operational base B will use the data from base B partition and the agents on base A will use the data from base A partition. The only exception is the *Solution Applicator* agent (AP) which has the authorization to register the solutions on both data partitions.

Finally, the *Physical Distribution* is obtained by distributing the agents and the environment by three different machines that communicate through an *Ethernet* connection. One server for base B, another for base A and the third for the company database (environment).

Having a multiple-instance agent architecture (distributed or not) allows to show two important characteristics of the multi-agent system we have developed:

1. The *social-awareness* of the agents, i.e., *an awareness of other agents in the environment and knowledge about their behavior*, according to (Kazakov & Kudenko, 2001).
2. How the GQN protocol works having multiple agents with similar roles.

7.4.2.1 Social-Awareness

Regarding the *social-awareness*, the agents are aware of other agents with the same role in the environment through the services they announce on the *Directory Facilitator Agent* of the JADE framework. As it is possible to see in Table 7.1 the agents announce the services they provide through the *DFRegisterAgent* method and subscribe notifications of new services through the *DF-SubscribeAgent* method. To better understand how this works, let us use as an example the agents *MT:B* and *MT:A* in Figure 7.4.

These agents are responsible for monitoring the operational plan and to decide if a specific event will cause a problem that needs to be solved. The first thing this agent does is to register its services on the *Directory Facilitator Agent*. For that, they use the *DFRegisterAgent* method and provide the information presented in Table 7.2. As it is possible to see, the differences are on the

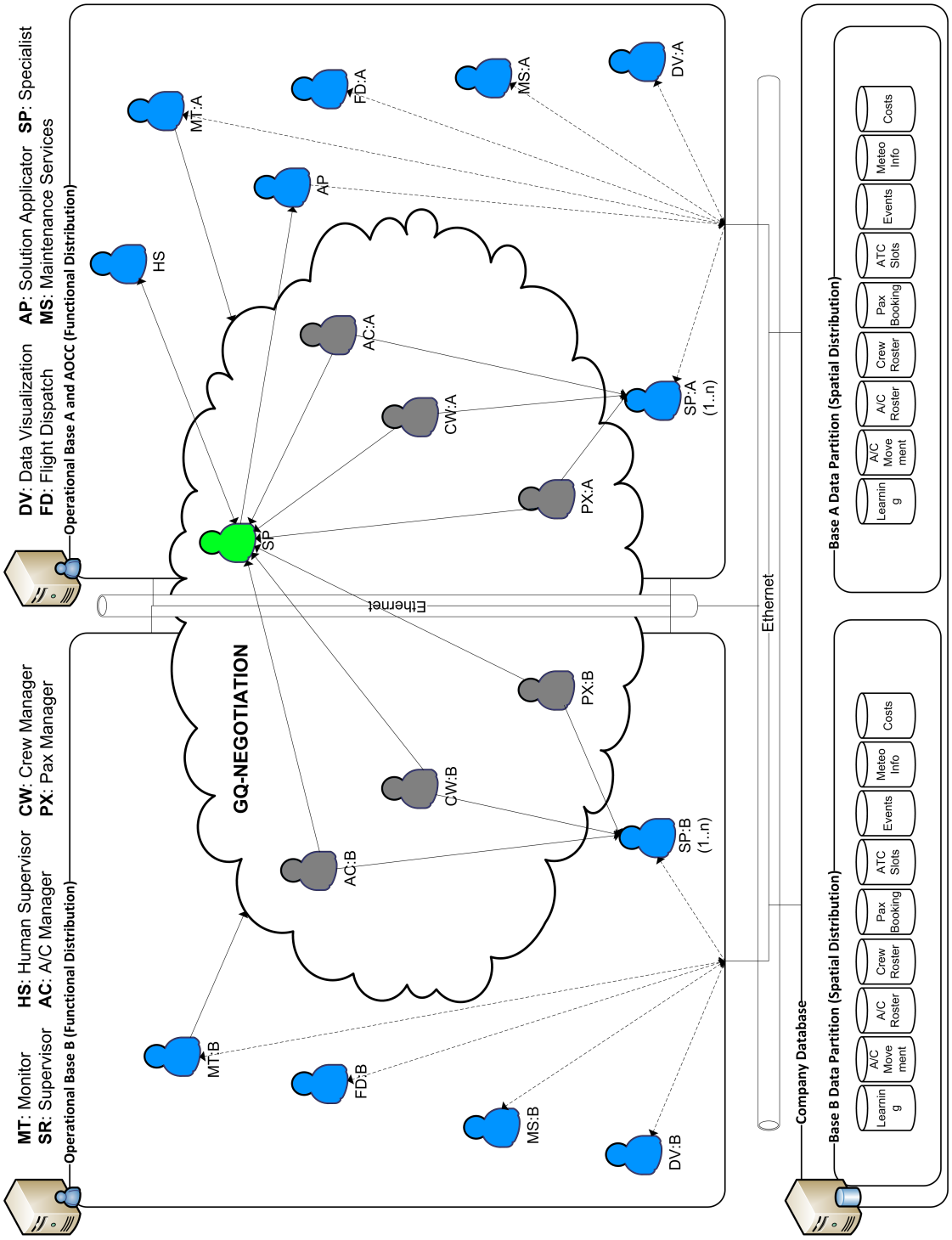


Fig. 7.4 Multi-Instance Architecture and Agents Interaction

Table 7.2 Properties necessary to register services for Monitoring agents

Properties	**Agent *MT:A***	**Agent *MT:B***
1:Service Name	monitoring-operational-plan	monitoring-operational-plan
2:Service Type	monitoring	monitoring
3:Ontology	monitoring-ops-plan-ontology	monitoring-ops-plan-ontology
4:Languages	fipa-acl	fipa-acl
5:operational-base	base-A	base-B
6:data-partition-name	base-A	base-B

properties *operational-base* and *data-partition-name*. These properties define the functional and spatial distribution of each monitor agent respectively.

Besides registering its services each monitor agent needs to subscribe notifications of new services that might be registered. For that they use the *DFSubscribeAgent* method and provide the properties indicated in Table 7.3. Now, what will happen when a new agent provides similar services? For example, let us suppose that a third monitor agent *MT:C* is registered on the system providing services for *operational-base = base-A and base-B* and *data-partition-name = base-A and base-B*. In this case the agents *MT:A* and *MT:B* will be notified of these services and since both provided services for *base-A* and *base-B* they will have to engage in an agreement in order to share their tasks. As an example, this might lead to the creation of a *base-C* by resizing *base-A* and *base-B*. This knowledge of other agents with similar role in the environment plus the knowledge about the services they provide is what we call *social-awareness*. The strategy built in each agent to split the tasks might be as simple as dividing it according to a specific criteria (e.g., between flight numbers or between time periods of the day) or any other more complex strategy that might include negotiation, for example.

Table 7.3 Properties necessary to subscribe notifications of new services

Properties	**Agent *MT:A***	**Agent *MT:B***
1:Service Type	monitoring	monitoring
2:operational-base	base-A	base-B
3:data-partition-name	base-A	base-B
4:max results	10	10

Finally, the agents also have the possibility to search for services provided by other agents through the *DFSearchAgent* method. For that, they just need to search for the *Service Type* they want (e.g., monitoring) and choose the one they are interested in (remember that each agent can provided more than one service as it is possible to see in the case of agent *MT:C*).

Although the example provided was about the monitor agents *MT:A* and *MT:B* a similar approach is followed by the other agents with the exception of the specialist agents in both operational bases (*SP:A* and *SP:B*) and the agents' supervisor *SP*, human supervisor *HS* and solution applicator *AP*. The specialist agents work under the direction of their managers who have social-awareness characteristics. Because of that, they do not need to also have that characteristic. The *SP* and *HS* agents make decisions regarding the all of the system, so, there is no need to have agents with similar roles. The same happens regarding the *AP* agent. It is important to point out that this is in the case of the functional distribution as presented in Figure 7.4. In a completely different

distribution it is perfectly possible to have more that one instance of the *SP*, *HS* and *AP* agents and, in that case, they will have *social-awareness* characteristics.

7.4.2.2 GQN Behavior

Regarding the *GQN protocol behavior* when faced with multiple agents with the same role, let us see how it works in the architecture presented in Figure 7.4. When an event is detected by agent *MT:A* or *MT:B* and that event causes a problem (i.e., a flight delay) the monitor agent notifies agent supervisor *SP* to find a solution. Let us suppose that it was agent *MT:B* that found a problem that affected a flight of operational base B. Agent *SP* will prepare a call-for-proposal and sends it to all managers in all operational bases, that is, agents *AC:A*, *CW:A*, *PX:A* and *AC:B*, *CW:B* and *PX:B*. Each one of these agents will prepare proposals considering the knowledge they have, i.e., agents from operational base B will use information and resources from base B and agents from operational base A will use information and resources from base A. As required by the interaction protocol (see Chapter 6 Section 6.5) each manager will present a candidate solution that includes the three dimensions of the problem and, for that, they will engage in a negotiation with the respondent agents that belong to the same operational base. For example, the agents *AC:B*, *CW:B* and *PX:B* will negotiate with each other but not with agents *AC:A*, *CW:A* and *PX:A*. The opposite is valid for the agents on operational base A.

Each one of these agents will send a proposal (if they choose to do so) meaning that the supervisor agent *SP* will receive six candidate solutions. The *SP* will choose the best proposal according to its preferences (trough a Utility Function - see Definition 6.12), informing the winner agent (see Definition 6.13) and giving feedback to the others (see Definition 6.5). From this point on the protocol goes as expected (more information about the protocol is given in Chapter 6 and in Section 7.6 we detailed how the protocol works when applied to the disruption management case). What is important to retain is that it *does not matter how many instances of the manager agents we have and their functional and spatial distribution* for the protocol to work properly. If the manager agents want to propose a candidate solution they need to include all the dimensions and the supervisor *SP* will choose the best one according to its preferences.

7.5 Operational Costs

In the previous section we have detailed the architecture of the system. Before proceeding to understand what are the decision mechanisms used in the system, it is important to understand what costs are involved when dealing with a disruption management scenario in airline operations control, i.e., what are the costs involved in a solution to a disruption.

Typically, in the air transport domain, the word *operational cost* means all the costs involved in the operation of a flight. For us, the *Operational Cost (OC)* of a specific solution for a disruption includes *Direct Operational Costs (DC)* and *Quality Operational Costs (QC)* and is given by Equation 7.1.

$$OC = DC + \beta \times QC \qquad \beta \in \Re, \quad \beta \geq 0 \tag{7.1}$$

Coefficient β is used to define the weight of quality costs. A detailed presentation of the several components of the direct operational costs and quality operational costs follows.

7.5.1 Direct Operational Costs

Direct Operational Costs (DC) of a specific solution for a disruption are costs that are easily quantifiable and are related to the operation of the flight (or flights included in the solution), namely, *Flight Costs (FC)*, *Crew Costs (CC)* and *Passenger Costs (PC)*. It is given by Equation 7.2.

$$DC = FC + CC + PC \tag{7.2}$$

7.5.1.1 Flight Costs

The *Flight Cost (FC)* for all flights included in a solution is given by Equation 7.3 and includes six types of costs, i.e.,

$$FC = \sum_{i=1}^{|Fl|} (TkOff_i + Land_i + Park_i + Hand_i + Maint_i + Atc_i + Fuel_i) \tag{7.3}$$

Fl is the set of all flights in the solution with $i \in \boldsymbol{Fl}$.

- *Takeoff charges* (*TkOff*): The airports usually apply a charge for each aircraft takeoff. The calculation method for the charge value is different for each airport. Usually, it is related to the MTOW (Maximum Takeoff Weight) of the aircraft paying more the ones that have a greater MTOW.
- *Landing charges* (*Land*): Similar to the previous one but applied for each aircraft landing. The charge value is also related to the MTOW of the aircraft.
- *Parking charges* (*Park*): The airports usually applies a charge for parking the aircraft. The charge value takes into consideration the time the aircraft is parked as well as the parking place (parking spaces further away from the terminal pay less). The method of calculation is usually different for each airport.
- *Handling charges* (*Hand*): This includes the many service requirements of an airliner between the time it arrives at a terminal gate and the time it departs on its next flight. For example, cabin services like cleaning, catering, ramp services like luggage handling and passenger stairs and passenger services like check-in and gate arrival and departures services. The value depends on the handling agent and it might be different in each airport.
- *Maintenance costs* (*Maint*): Includes the line maintenance (i.e., the maintenance that might need to be performed during the turn-around of the aircraft at the airport) as well as aircraft maintenance period checks. The values are different for each airline depending if the airline has its own maintenance services or not.
- *Air Traffic Control charges* (*Atc*): Two main group of charges are included: *en-route* and *terminal navigation* charges. The former are charges related to air traffic services during the route of an aircraft. The value is calculated according to the MTOW of the aircraft and the distance flown. In Europe the EUROCONTROL Central Route Charges Office (CRCO) bills and collects route charges on behalf of all EUROCONTROL's Member States. On the other countries the charges and method of calculation depends on the specific country policy. The latter are charges related to air traffic services in a terminal control area, control zone or flight information zone of an airport. The calculation method of these charges usually depend on each country policy.

- *Fuel costs* (*Fuel*): Includes the costs with fuel to go from the origin to the destination (*trip fuel*) plus any additional extra fuel required. The values are different for each airline and depends on the contracts that each airline has with the fuel companies.

Each airport charges an airline for *Takeoff*, *Landing* and *Parking* of an aircraft. The manual *Airport and Air Navigation Charges Manual* (IATA, 2001) from IATA provided us the method we have used to calculate the values. Some simplifications were made because the amount of variables and differences between airports is overwhelming. We used average values of MTOW and applied the fee charged by each airport per tonne of MTOW to two categories of aircraft: Wide Body (WB) and Narrow Body (NB).

Table C.1 in Appendix C presents the values for takeoff, landing and parking, per airport and aircraft fleet type (i.e., WB and NB). If we consider the table values in absolute terms then, they do not correspond to reality. They are only an estimation. However, because the simplification method used was the same for all airports, we can make comparisons between them because the order of magnitude was observed.

Example

Imagine flight TP442 from Lisbon (LIS) to Orly (ORY) with a Narrow Body (NB) aircraft, arriving to ORY at 10 am. and having the next departure from ORY at 12.30 pm. Using the Table C.1 the value for take off (*TkOff*) in Lisbon is 374,939 m.u. and the value for landing (*Land*) in Orly is 460,423 m.u. Regarding parking (*Park*), Table C.1 shows values for daily parking (24 hours). So, in this example, the aircraft would be 150 min parked at ORY. Calculating the charge for this parking time for NB aircraft we have *Park* = 618,398 m.u. / 1440 min x 150 min = 64,416 m.u.

As with the other airport charges, *Handling* charges are calculated differently from airport to airport. We adopted the model and costs used by the Macedonian *Alexander The Great* airport because it is the simplest one. This is a one time fee charge per flight according to the aircraft model. The values used are presented in Table C.3, column *Airport Handling per aircraft* in Appendix C. For example, let us assume that the aircraft model of the example above is an Airbus A319. The handling cost for that flight (*Hand*) is 950 m.u.

Regarding the *Maintenance* costs we have defined an average value per minute and aircraft model, using as the source of information a study from the *Transportation Studies Group* of the University of Westminster (Group, 2008). The values used are presented in Table C.3, in the column *Maintenance Cost per minute* in Appendix C. Continuing with the example above and assuming that the flight time from LIS to ORY takes 140 min the maintenance costs for that flight (*Maint*) is 140 min x 10,5 m.u = 1.470 m.u.

The Air Traffic Control (ATC) charges are applied by aircraft model and per nautical mile (nmi). The value per aircraft and nautical mile were provided by TAP Portugal and are presented in Table C.3, column *ATC Cost per nmi* in Appendix C. To be able to calculate the ATC charges of a flight we need to know the distance between the departure and arrival airports. Considering the example, we can see in Table C.2 that the distance between LIS and ORY is 776,29 nmi. So, the ATC costs for the flight is 776,29 nmi x 2,38 m.u. = 1847,57 m.u.

Finally, regarding *Fuel* costs we also used the information provided by TAP Portugal as presented in Table C.3, column *Fuel Cost per minute*. Continuing with our example we know that the flight took 140 min and the fuel cost per minute for the Airbus 319 is 15,34 m.u., so, *Fuel* is 140 min x 15,34 m.u = 2.147,6 m.u.

Summarizing the example we will get the following calculations and result for the TP442 flight costs:
$FC_{TP442} =$
$\sum_{i=1}^{1}(374,939 + 460,423 + 618,398 + 950 + 1.470 + 1.847,57 + 2.147,6) = 7.868,93\ m.u.$

7.5.1.2 Crew Costs

The *Crew Cost (CC)* for all flights included in a solution is given by Equation 7.4 and includes five types of costs, i.e.,

$$CC = \sum_{i=1}^{|Fl|}\sum_{j=1}^{|Cw|}(Salary_{i,j} + ExtTime_{i,j} + Perdiem_{i,j} + Hotel_{i,j} + Dhc_{i,j}) \tag{7.4}$$

Fl is the set of all flights in the solution with $i \in \boldsymbol{Fl}$ and ***Cw*** is the set of all crew members in the flight with $j \in \boldsymbol{Cw}$.

- *Salary costs* (*Salary*): The salary costs of all crew members of a flight. The salaries vary according to the crew members salary rank.
- *Extra Time costs* (*ExtTime*): In each monthly salary a specific number of monthly and annual duty and flight hours are included, i.e., there is a monthly and annual hours *plafond*. If a crew members exceeds the *plafond*, additional work hours are to be paid by the company according to the specific crew member salary rank hour value.
- *Perdiem costs* (*Perdiem*): For each duty day the crew members receive a specific allowance. The value of the allowance also varies according to the salary rank.
- *Hotel costs* (*Hotel*): It is common for the crew members to have flight duties that include a night stop or a layover for more than one day in specific cities. The airline company supports the costs associated with the stay at the hotel during the night stop or layover. The costs vary depending on the contract that the airline company has with each hotel.
- *Flight as Passenger costs* (*Dhc*): It is common for the crew members to travel as *extra-crew*, i.e., although not working on that flight they are traveling to another place to start or end their flight duty. If the extra-crew takes place in a flight from another company there are some costs associated.

The values we used to calculate the *Salary*, *Extra-Time* and *Perdiem* costs come from TAP Portugal and are presented in Table C.6 in Appendix C. These values are different depending on each crew member's salary rank. We have calculated these three types of costs for a specific crew member as follows:

- *Salary* cost: $HourlySalary \times ServiceHoursFlight$. The service flight hours includes the briefing time (usually 60 minutes before flight departure) and the debriefing time (usually 30 minutes after arrival of the last flight leg).
- *Extra-time* cost: $HourlySalary \times ExtraHoursOfDuty$. The *ExtraHoursOfDuty* can be from exceeding the monthly and/or the yearly *plafond*.
- *Perdiem* cost: $Perdiem \times NumberOfDutyDays$. For each day of duty the airline company pays a perdiem as an allowance to each crew member.

Some of the flight duties performed by the crew members require a night stop or layover in specific countries and cities. It is the responsibility of airlines to book and pay for accommodation of crews in hotels. For that, each airline has contracts with hotels all over the world, so that they can have a better price for crew accommodation. Typically, the hotel for each crew member is calculated as $NumberOfNights \times HotelChargePerNight$. In our study we have used the information about hotel crew costs provided by TAP as presented in Table C.5, column *Crewmember Cost per night* in Appendix C.

Finally, it is common for airlines to assign crew members to fly as a passenger on a specific flight so they can get to another city to work where they will pick up their assigned trip sequence, or to return to their operational base after performing a flight. This is known as *extra-crew* or *Dead head crew*. If they are flying in a flight of the same company, usually there is no cost associated. However, if they are flying in another company, typically there is a cost. We have used TAP Portugal costs according to Table C.4 in Appendix C.

7.5.1.3 Passenger Costs

The *Passenger Cost (PC)* of the *delayed passengers* for all flights included in a solution is given by Equation 7.5 and includes three types of costs, i.e.,

$$PC = \sum_{i=1}^{|Fl|} \sum_{d=1}^{|Pxd|} \left(Meals_{i,d} + Hotel_{i,d} + Comp_{i,d} \right) \tag{7.5}$$

Fl is the set of all flights in the solution with $i \in$ ***Fl*** and ***Pxd*** is the set of all delayed passengers in the flight with $d \in$ ***Pxd***.

- *Meal costs* (*Meals*): These are the costs associated with meals provided at the airport for passengers of flights that are delayed for several hours or canceled.
- *Hotel costs* (*Hotel*): These are the costs associated with accommodation for the passengers when a flight is canceled or delayed for several hours.
- *Compensation costs* (*Comp*): Besides the preceding two types of costs the passengers might have the right to receive compensations for delays and/or cancellations according to regulations.

The values we have used to calculate these costs result not only from the information received from TAP Portugal but, also, from the regulation of the European Union, specially, the *Regulation (EC) 261/2004* (Parliament, 2004).

Regarding the *Meal* costs we have used a fixed value of 25 m.u. for each passenger meal that needs to be paid. This value is charged every time a flight is delayed for more than two hours. For the *Hotel* cost we have used the information provided by TAP as presented in Table C.5, column *Passenger Cost per night* in Appendix C and we apply this cost for every passenger of every flight that is delayed for more than twelve hours.

Finally, regarding the *Compensation* costs we have used the values as defined in *Article 7 - Right to Compensation* of the *Regulation (EC) 261/2004* (Parliament, 2004). These values are different according to the flight distance and are as follows:

- 250 m.u. for flights of less than 1500 km.
- 400 m.u. for flights of more than 1500 km and less than 3500 km
- 600 m.u. for flights of more than 3500 km

7.5.2 Quality Operational Costs

We define *Quality Operational Costs (QC)* of a specific solution for a disruption as the costs that are not easily quantifiable and are related to the perception of the passenger, of the delay time and/or cancellation of a flight. That is, we try to quantify in a monetary unit the cost that each minute of delay and flight cancellation has from a passenger's point of view and, at the same time, use that value together with the *Direct Operational Costs* to improve the decision making during the disruption management process as presented on Equation (7.1). A term with a very similar meaning is *Passenger Goodwill*. In this section we propose an innovative method of capturing and measure this very important cost component.

7.5.2.1 Perception of Quality Costs

It is generally accepted that, the best solution to a disruption in the AOC domain, is the one that does not delay the flight and has the minimum direct operational cost. Unfortunately, due to several reasons (see (Kohl & Karisch, 2004) for several examples as well as Section 4.5), it is very rare to have candidate solutions that do not delay a flight and/or do not increase the direct operational cost. From the observations we have done on TAP's AOCC, most of the times, the team of specialists has to choose between candidate solutions that delay the flight and increase the direct operational costs. Reasonably, they choose the one that minimize these two values.

We also found that some teams in the AOCC used some kind of *rule-of-thumb* or *hidden knowledge* that, in some cases, incites them not to chose the candidate solutions that minimize the delays and/or the direct operational costs. For example, suppose that they have disruptions for flight A and B with similar schedule departure time. To solve the problem, they have two candidate solutions: one is to delay flight A 30 minutes and the other would delay flight B 15 minutes. The direct operational costs for both candidate solutions are the same. Sometimes they would choose to delay flight A 15 minutes and flight B 30 minutes. We can state that flights with several business passengers, VIP's or for business destinations correspond to the profile of flight A in the above example. In our understanding this means that they are using some kind of quality costs when making decisions, although not quantified and based on personal experience. In our opinion, this knowledge represents an important part in the decision process and should be included in it. From our observations, this is not usually the case and, as such, our proposal is a contribution of ours to a more credible costs function.

7.5.2.2 Quantifying Quality Costs

To be able to use this information in a reliable decision process we need to find a way of quantifying it. What we are interested in knowing is how the delay time and the importance of that delay to the passenger are related in a specific flight. It is reasonable to assume that, for all passengers in a flight, less delay is good and more is bad. However, when not delaying is not an option and the AOCC has to choose between different delays to different flights which one should they choose? We argue that the decision should take into consideration the specific flight passenger's profile and not only the delay time and/or operational cost. For quantifying the costs from the passenger's point of view, we propose the following generic approach:

1. Define the existing passenger's profile(s) in the flight.
2. Define a delay cost for each passenger in each profile.
3. Calculate the quality costs using the previous steps.

Most likely, every airline company will have a different method to *define the passenger profile* in a specific flight. Most of the airlines will just consider one or two profiles (for example, business and economy). To get the number of passengers that belong to these profiles is very easy. Airline companies can use the flight boarding information to calculate this number complemented with some very simple questions that can be asked during the check-in/boarding procedure. For example, the reason for traveling (a typical answer could be: business, pleasure or both). Using an appropriate information system it will be possible to obtain this information dynamically and for the specific flight(s) involved in the disruption.

Most of the airline companies will choose to use a fixed *delay cost value for each passenger of each profile*. These numbers can reflect the perception of the costs from the point of view of the company or can result from a statistical analysis of the company information. In our opinion and that is one of the *main contributions of our approach*, we think that this cost should be calculated from the passenger's point of view. This implies the use of a formula to calculate each profile cost, that represents this relation. In section 7.5.2.3 we show how a real airline company used a passenger survey to derive three passenger profiles and obtain formulas to calculate the cost of each passenger profile. Given the above, we believe that the quality costs of a flight should result from the relation between the number of passenger profiles in that flight and the delay cost for each passenger with those profiles from their point of view, expressed by the Equation 7.6.

$$QC_f = \alpha \sum_{p=1}^{|PP|} (P_p \times C_p) \tag{7.6}$$

In this equation α is a coefficient that can be used to increase or decrease the order of magnitude, $\boldsymbol{C}$ represents the delay cost (in monetary units) of each passenger on profile $\boldsymbol{p}$ and $\boldsymbol{P}$ represents the number of passengers of profile $\boldsymbol{p}$. Additionally, $p \in PP$ (PP is the set of all passengers' profiles in a flight) and $f \in Fl$ (Fl is the set of all company flights).

The delay cost should be related to the minutes of flight delay and the number of passengers on the profile and is given by Equation (7.7). In this equation pl represents the number of passengers that may lose the flight connection (when compared with the expected delay time) and tp the total number of passengers on the profile. The γ coefficient allows the increase of importance of these passengers that may lose the connection and should be ≥ 1.

$$P_p = (\gamma \times pl + tp) \tag{7.7}$$

In our opinion, the Equations (7.6) and (7.7) are generic enough to be used according to the different models and policies that airlines have to quantify the passenger's goodwill. For example, to have a fixed monetary value for each passenger independently of the number of profiles and/or to have a fixed monetary value for each minute of delay. It also allows more sophisticated ways of calculating the monetary value to each minute of delay.

7.5.2.3 Airline Company Case Study

In this section we are going to show how we used our method to obtain passenger profiles and costs to every flight for TAP Portugal, regarding the *delay cost from the point of view of the passengers*. To get this information, we have done a survey of several passengers on flights of the airline company. Besides asking in what class they were seated and the reason for flying in that specific flight, we asked them to evaluate from 1 to 10 (1 - not important, 10 very important) the following delay ranges (in minutes): less that 30, between 30 and 60, between 60 and 120, more than 120 and flight cancellation. From the results we found the passenger profiles in Table 7.4.

Table 7.4 Passenger Profiles

Profiles	Main Characteristics
Business	Travel in first or business class; VIP; Frequent Flyer members; Fly to business destinations; More expensive tickets.
Pleasure	Travel in economy class; Less expensive tickets; Fly to vacation destinations.
Illness	Stretcher on board; Medical doctor or nurse traveling with the passenger; Personal oxygen on board or other special needs.

For the profiles in Table 7.4 to be useful, we need to be able to get the information that characterizes each profile, from the airline company database. We found that we can get the number of passengers of each profile in a specific flight from the boarding database, using the information in Table 7.5.

Table 7.5 Boarding Information

Profiles	Relevant Attributes for Profiling
Business	Number of passengers in business class; Number of VIPs; Number of Frequent Flyer passengers; Number of passengers according to the ticket price and Departure or Arrival for or from a business location.
Pleasure	Number of passengers in economy class; Number of passengers according to the ticket price and Departure or Arrival for or from a vacation location.
Illness	Number of passengers with special needs; Stretcher on board.

Besides being able to get the number and characterization of profiles from the survey data, we were also able to get the trend of each profile, regarding delay time/importance to the passenger. Plotting the data and the trend we got the graph in Figure 7.5. If we apply these formulas as is, we would get quality costs for flights that do not delay. Because of that we re-wrote the formulas. The final equations that express the importance of the delay time for business, pleasure and illness passenger profile are Equations 7.8, 7.9 and 7.10, respectively.

$$BC_f = 0.16 * d_f^2 + 1.38 * d_f \tag{7.8}$$

$$PC_f = 1.20 * d_f \tag{7.9}$$

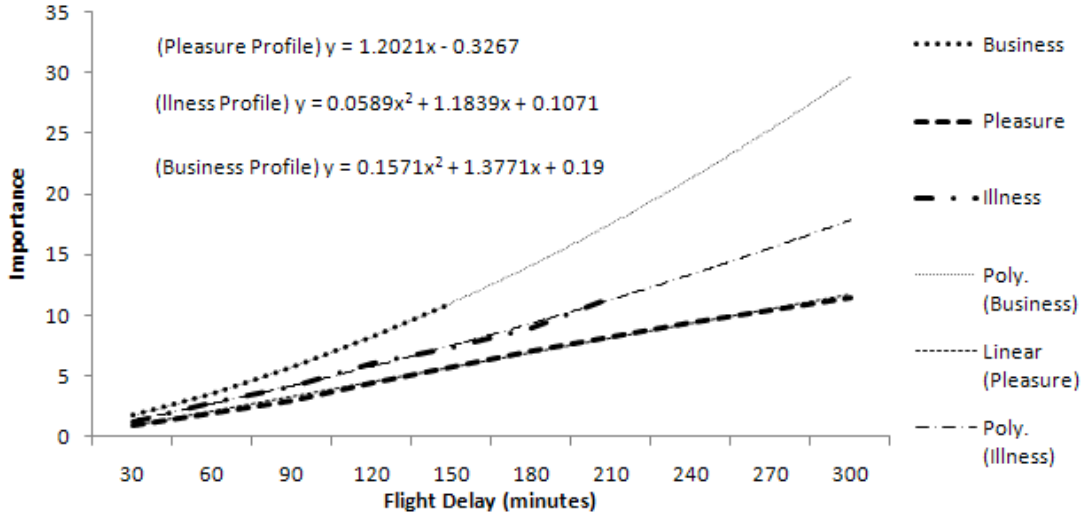


Fig. 7.5 Case Study Trend Formulas for the Profiles

$$IC_f = 0.06 * d_f^2 + 1.19 * d_f \tag{7.10}$$

In these equations, $\boldsymbol{f}$ represents a specific flight from the set of flights of the airline and $\boldsymbol{d}$ the delay time (in minutes) of flight $\boldsymbol{f}$.

It is important to point out that these formulas are valid only for this particular case and only expresses the information we have from this specific survey data. When applied permanently as a method to calculate quality costs, the profiles and formulas can be updated every year, using the annual company survey, and obtaining different formulas according to flight destinations, flight schedules and/or geographical areas, by month, day, hour, etc. When combined with information from the booking and boarding systems, it is possible to get updated profiles and formulas more often. For example, by making a simple question during the check-in to each passenger (e.g., what is the reason for traveling?) and using information obtained through previous surveys and statistical data, it is possible to derive almost in real-time the number of profiles and profile costs for a specific flight.

Application Example

Let's calculate the quality operational costs for the following flight (assuming 2 as the order of magnitude coefficient): Flight 103 will be delayed 30 minutes at departure. It has 20 passengers in the *business* profile *(BC)*, 65 in the *pleasure* profile *(PC)* and 1 in the *illness* profile *(IC)*. If nothing is done to reduce the delay, 2 passengers from *BC* will lose their flight connections as well as 10 from the *PC* profile. Applying the formulas presented in Equation 7.8, 7.9 and 7.10, the cost of 30 minutes delay for each passenger in each profile is:

$$BC_{103} = 0.16 * 30^2 + 1.38 * 30 = 185.4\ m.u.$$

$$PC_{103} = 1.20 * 30 = 36\ m.u.$$

$$IC_{103} = 0.06 * 30^2 + 1.19 * 30 = 89.7\ m.u.$$

Additionally and considering the passengers that may lose the flight connection we have the following number of passengers per profile (according to Equation (7.7)):

$$P_{BC} = 2 + 20 = 22$$

$$P_{PC} = 10 + 65 = 75$$

$$P_{IC} = 0 + 1 = 1$$

The quality operational cost for the flight 103 (QC_{103}) according to Equation 7.6, is:

$$QC_{103} = 2 \times (22 \times 185.4 + 75 \times 36 + 1 \times 89.7) = 13737\ m.u.$$

7.6 Decision Mechanisms and Learning

In the previous sections of this chapter we have described the MAS architecture and organization, including the roles and functions of each agent in the system. Additionally, we have described the operational costs involved when a disruption happens. Now, it is time to explain how the agents in our MAS make decisions about the best solutions to apply in a specific disruption.

As we have described in Section 3.4 (*Airline Operations Recovery*) and as we have explained in Chapter 4 (*The Airline Operations Control Problem*), the majority of the current work of other researchers, typically follow a sequential approach when solving a disruption. Figure 7.6 shows that in the sequential approach, the *aircraft* dimension of the problem is solved first, then, using the aircraft partial-solution as the input, the *crew* dimension is solved (in the case of not having a crew available, another aircraft partial-solution is used as input) and, finally, using the aircraft and crew part of the solution as the input, a solution for the *passenger* dimension is found.

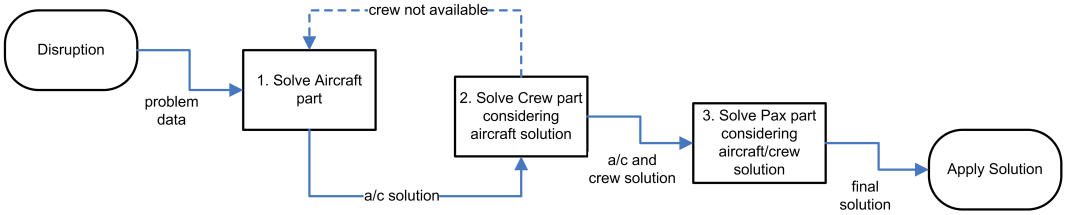


Fig. 7.6 Typical Sequential Approach

In our opinion using this sequential approach a natural order of importance is imposed to each dimension of the problem, i.e., the aircraft dimension is more important than the crew dimension and both are more important than the passenger dimension. We believe that to get a better integrated solution, each dimension must have the opportunity to be at the same level of importance. By *opportunity to be at the same level of importance* we mean that the agent that represents a dimension should be able to generate and present full candidate solutions (i.e., solutions that include the three dimension of the problem), from their local point of view and considering its preferences

and interests. It should be from this set of full candidate solutions that the best one according to the global interest of the AOCC is chosen.

Considering the above we have adopted automated negotiation as a mechanism for our agents to reach a decision and, as such, give the *opportunity* for all dimensions of the problem *to be at the same level of importance*.

In Chapter 6 we have proposed an automated negotiation protocol called GQN (*Generic Q-Negotiation*) that has the necessary properties to be used in the new approach for disruption management we are proposing in this chapter. Although we have explained in detail how the protocol works, including how we have applied it to model two different application examples, here we are going to show how we have used this protocol as a decision mechanism in MASDIMA (see Appendix A), considering the MAS architecture as presented in Figure 7.2. Our goal in this section is to focus on the decision making and learning characteristics of the protocol as applied in this specific case, giving more details about how the supervisor chooses the best proposal from the ones received from the managers and how each manager chooses from the candidate-solutions the best one to send in a proposal.

MASDIMA uses two levels of negotiation, the *Manager Agents Level*, i.e., negotiation that involves the *supervisor*, *A/C Manager*, *Crew Manager* and *Pax Manager* agents and the *Team Agents Level* involving each manager and its teams of specialist agents. Both levels as well as the learning characteristics used in the first one, will be explained on the next sections.

7.6.1 Manager Agents Level

Figure 7.7 shows an overview of how the GQN negotiation protocol is applied in MASDIMA. The full sequence diagram of the protocol is presented in Appendix B, Figure B.1.

The *Monitor* agent sends the problem to the *Supervisor* agent, including information about the event that triggered the problem (e.g., aircraft malfunction or bad weather) and the dimensions (e.g, the aircraft, flight, crew and passengers involved) as well as the schedule time and schedule costs. The agent Supervisor (that assumes the role of *organizer*) using this information, prepares a call-for-proposal (cfp) that includes the problem according to definition 6.1 and a negotiation deadline and sends the *cfp* to the *A/C*, *Crew* and *Passenger* managers, at the same time. An example of a *cfp* sent to agent *Crew* is $\langle supervisor, crew, 1, P, 15\rangle$. In the case of MASDIMA the *Problem* includes three dimensions, i.e.,

$$P = \langle d_{a/c}, d_{crew}, d_{pax}\rangle \tag{7.11}$$

and the dimensions include the following attributes (here we are not showing the private components of the dimensions):

$$d_{a/c} = \langle airc, \{ad, ac\}, \{\aleph, \Re\}\rangle \tag{7.12}$$

$$d_{crew} = \langle crew, \{cd, cc\}, \{\aleph, \Re\}\rangle \tag{7.13}$$

$$d_{pax} = \langle pax, \{pt, cc\}, \{\aleph, \Re\}\rangle \tag{7.14}$$

Regarding the attributes, *ad* is the aircraft delay, *ac* the aircraft cost, *cd* is the crew delay, *cc* the crew cost, *pt* is the passenger trip time (i.e., the elapsed time between the departure of the first flight of the passenger and the arrival of the last flight) and *pc* the passenger cost.

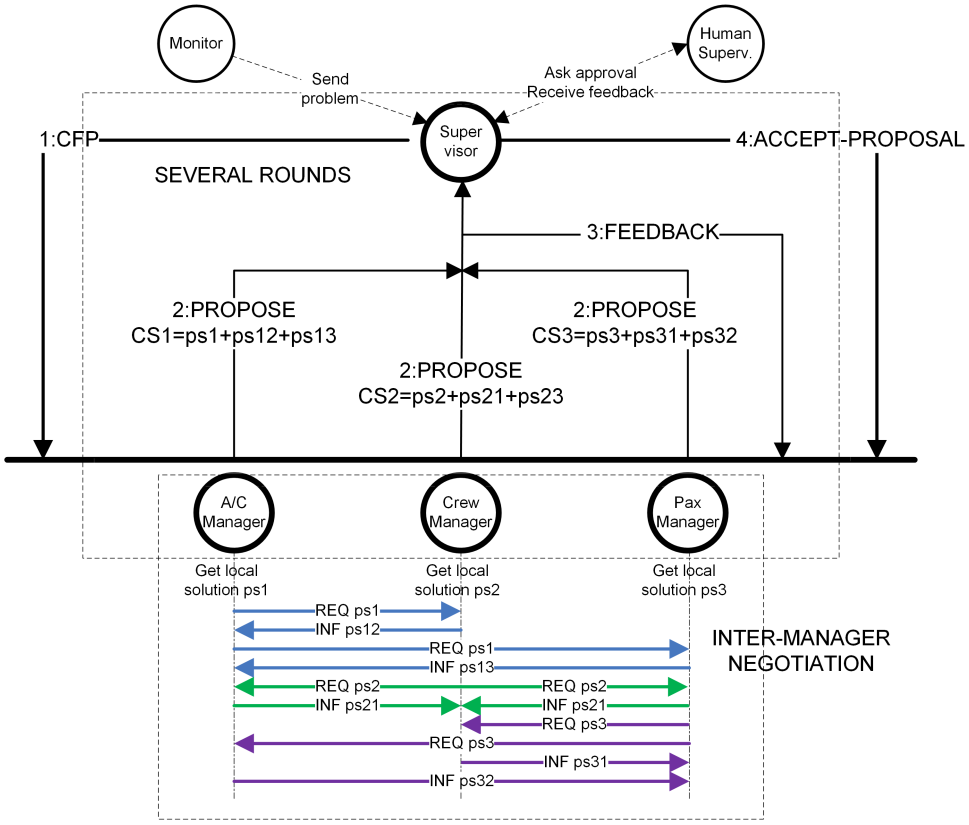


Fig. 7.7 GQN Negotiation Protocol Applied in MASDIMA

After the call-for-proposal, the first round of negotiation starts. The *A/C*, *Crew* and *Passenger* managers present the proposal according to their interests expressed by an utility function. The *A/C* manager uses the Equation (7.15) as the utility function.

$$U_{a/c} = 1 - \left(\frac{w_1 \left(\frac{ad}{max(ad)} \right) + w_2 \left(\frac{ac}{max(ac)} \right)}{w_1 + w_2} \right) \tag{7.15}$$

In this equation the aircraft or flight costs are calculated according to subsection 7.5.1.1. The w_1 and w_2 represents the importance of attribute *ad* and *ac*, respectively, and each one must have a value between 0 and 1. This means that agent *A/C* prefers solutions that minimize the aircraft delays and flight costs.

Likewise, the *Crew* manager uses Equation (7.16) as the utility function.

$$U_{crew} = 1 - \left(\frac{w_3 \left(\frac{cd}{max(cd)} \right) + w_4 \left(\frac{cc}{max(cc)} \right)}{w_3 + w_4} \right) \tag{7.16}$$

The crew costs are calculated according to subsection 7.5.1.2. This means that the *Crew* agent prefers solutions that minimize the crew delays and crew costs.

Finally and similar to the previous ones, the *Passenger* manager uses Equation (7.17) as the utility function.

$$U_{pax} = 1 - \left(\frac{w_5 \left(\frac{pt}{max(pt)} \right) + w_6 \left(\frac{pc}{max(pc)} \right)}{w_5 + w_6} \right) \tag{7.17}$$

Here the passenger costs is calculated according to subsection 7.5.1.3.

It is important to point out that these utility functions (i.e. the preferences) are private to each manager, meaning that only its owner knows about it. Each manager requests candidate solutions to its team of specialist agents and orders them according to the utility function in the first round complemented by the use of the Q-Learning algorithm as explained in Section 7.6.3 on subsequent rounds.

In the AOCC scenario and according to the MASDIMA architecture (Figure 7.2), the manager agents do not have the knowledge to present proposals that include the three dimensions, i.e., each agent has only a *competence* (Definition 6.11) on a dimension. Because of that, the manager agents need to engage in a *Inter-Manager Negotiation* to be able to prepare a proposal to send to the Supervisor.

Looking at Figure 7.7 the *A/C Manager* agent gets a list of local partial-solutions (i.e. a list of solutions for the aircraft dimension) according to its preferences. Using the first one, *ps1* (starting with the second round, the choice of the partial-solution to use is done according to the Q-Learning algorithm) and assuming the role of *initiator*, sends a request to agents *Crew* and *Pax* (these agents assume the role of *participant*), simultaneously, asking them to provide their best local crew and passenger partial-solutions (*ps12* and *ps13*, respectively), considering the *ps1* and complying with the restrictions sent. An example of a request sent to the Crew manager is

$$request = \langle a/c, crew, \theta, \phi_{crew} \rangle$$

The symbol θ here represents a serializable object that includes the *ps1* partial-solution as well as a simple text statement like *"requesting crew partial-solution to aircraft ps1 partial-solution"*.

A restriction sent to the Crew manager by the A/C manager could be

$$\phi_{crew} = < ad \leq 15 > \wedge < crewFleet = NB >$$

This restriction means that the Crew manager should send a partial-solution for the crew dimension that does not delay the flight more than 15 minutes and with a crew that belongs to the *NB* (Narrow Body) fleet. For the Passenger manager and supposing that we have one disrupted passenger, an example could be

$$\phi_{pax} = < pt \leq 15 > \wedge < availSeats \geq 1 > \wedge < origAirp = LIS > \wedge < destAirp = ORY >$$

In this case, the passenger partial-solution should not delay the passenger trip time for more than 15 minutes, the alternative itinerary should be in a flight with at least 1 seat available, starting at Lisbon and ending in Orly.

The Crew and Passenger partial-solutions are sent to the A/C manager, through an *inform* performative and the candidate-solution that is going to be part of the proposal is the aggregation of these three partial-solutions, i.e.,

$$CS1 = ps1 + ps12 + ps13$$

According to definition 6.14 this is a *Coherent Solution*.

Likewise, the Crew manager engages in a *Inter-Manager Negotiation* with *A/C* and *Pax* managers and the Pax manager with *A/C* and *Crew* managers, presenting candidate-solution *CS2* and *CS3* respectively.

During the *Inter-Manager* negotiation if the *participant* manager can not comply with the restrictions it will send a *failure* to the *initiator*. In this case, the *initiator* will send a new request using its second best local partial-solution and the above process is repeated.

The candidate-solutions included in the proposals received by the Supervisor agent are evaluated and ordered according to the utility function 7.18 and the *round winner* proposal is chosen according to Definition 6.13. To the manager agents that lost, qualitative feedback is sent according to Definitions 6.5 and 6.16.

$$U_{sup} = 1 - \left(\alpha_1 \left(v_1 \left(\frac{ad}{max(ad)} \right) + v_2 \left(\frac{ac}{max(ac)} \right) \right) + \alpha_2 (\ldots) + \alpha_3 (\ldots) \right) \tag{7.18}$$

with

$$\sum_{i=1}^{3} \alpha_i = 1 \; and \; \sum_{j=1}^{6} v_i = 1$$

The v_j represents the importance of each attribute in the dimension and α_i the importance of each dimension in the problem.

Using the feedback, the manager agents that lost change their proposals. Starting with the second round the proposal formulation process uses a Q-Learning algorithm endowing the agent with the capability to learn on-line along the process. This process is explained in Section 7.6.3. This loop of proposals and feedback rounds ends according to the *negotiation termination rule* as explained in Section 6.6, meaning that the Supervisor agent found a proposal that includes a candidate-solution that satisfies its preferences.

It is important to point out that the candidate-solution includes only the attribute values for each attribute of each dimension represented according to Definition 6.4. For example, $S = \langle 5, 6200, 0, 7100, 5, 1000 \rangle$ meaning that the *flight delay* is 5 minutes and the *flight costs* is 6200 m.u., the *crew delay* is 0 and the *crew costs* is 7100 m.u. and, finally, the *passenger trip time delay* is 5 minutes and the *passenger costs* is 1000 m.u. However, this representation encodes an *associated plan* that includes the actions to be performed on the operational plan to change it. For example, *exchange aircraft CS-TMW with aircraft CS-TNN from flight TP101*, *use reserve crew with id 12 to replace absent crew with id 23* and, finally, *keep the disrupted passengers in the same flight*. More details about this *associated plan* and possible actions is given in Sections 7.6.2 and 7.7.

The candidate-solution included in the *negotiation winner proposal*, the associated plan and the rationale behind it is sent to the *Human Supervisor* that can choose to approve it or not. If the human supervisor approves, the associated plan will be applied and the operational plan is changed accordingly. If he/she chooses to not approve the solution, some feedback is given. Using this feedback, the Supervisor agent will have to improve its preferences and utility function and issue a new call-for-proposal to restart the negotiation process. Since in the experiments we have performed we did not include the feedback provided by the human supervisor, we give more details about this features in the *Advanced Features* Section (7.8) of this chapter.

7.6.2 Team Agents Level

At the Team Level we use a fipa-contract.net (FIPA, 2002b; Smith, 1980) protocol with some modifications. Figure 7.8 presents this protocol applied to the *Crew Manager* team that we are going to use as an example to explain it, since it works the same way when applied to the *A/C* or *Pax* manager. In the *Crew Manager* team there are two *specialist* agents:

1. *CrewSimmAnneal*: this agent implements a Simulated Annealing (Kirkpatrick *et al.*, 1983) algorithm to look for solutions regarding the crew dimension.
2. *CrewHC*: this agent implements an Hill Climbing algorithm (Russell & Norvig, 2003) with the same goal as the previous one.

The details about the specialist agents are given in Section 7.7. Here we are going to focus on the way these agents interact with the manager agent.

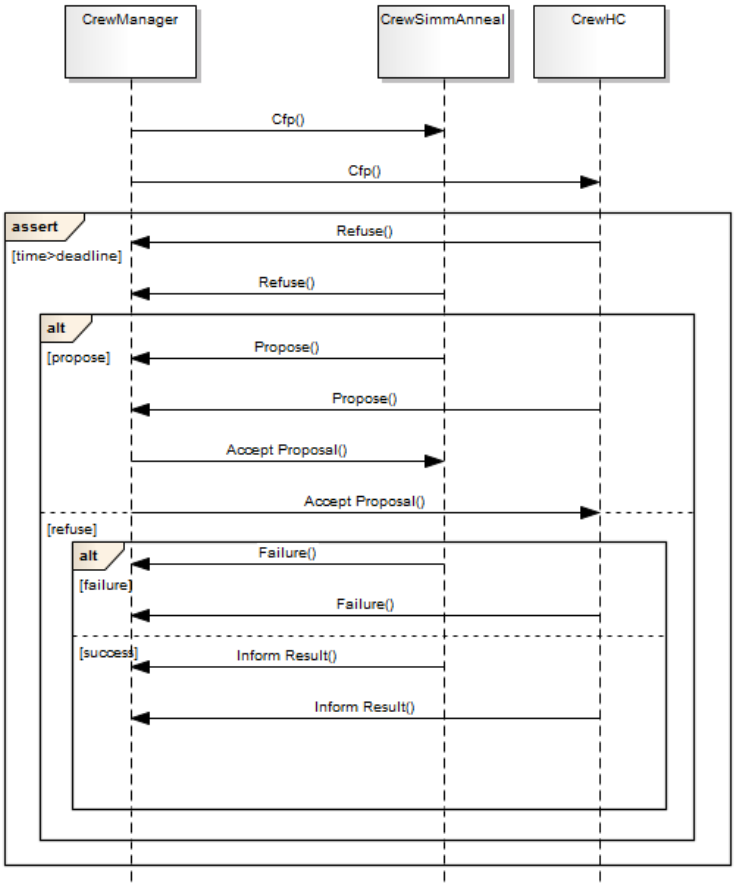


Fig. 7.8 Contract Net Protocol (simplified) Applied at Crew Team Level in MASDIMA

If the *Crew Manager* agent decides to propose a candidate-solution in a specific round to the Supervisor, it needs in the first place to get a list of partial-solutions for the dimension on which it has a competence (in this example, crew dimension). To generate the partial-solutions, the *Crew Manager* needs to have, at least, one specialist agent that implements an adequate problem solving algorithm. Having more than one, the problem solving algorithms should be different, otherwise they will produce the same partial-solutions and there is no advantage on that.

To start the *negotiation* with the specialist, the *Crew Manager* issues a *cfp* (call-for-proposal) to all specialist agents simultaneously. On the *cfp* it is included information about the problem as well as two deadlines: one for receiving an answer stating that the specialist agent is going to present or not partial-solutions and another deadline for receiving the partial-solutions from the specialist agent.

After receiving the *cfp* the specialist agents have two options: to *refuse* to participate or answer with a *propose* meaning that it will seek possible partial-solutions according to the *cfp* conditions. The *Crew Manager* answers back with an *accept-proposal*. To speed-up the communication, we have simplified the protocol here. In our approach, we do not need to select from the received answers because we want all available agents to work in parallel. That is the reason why the answer from the specialist agents is ”yes” or ”no”, meaning that they are available (or not) to seek for partial-solutions. If the specialist agent finishes the task with success, it will send the partial-solutions included in the *inform-result* performative. If it fails, the reasons are included in a *failure* performative.

After receiving all the partial-solutions, the *Crew Manager* agent needs to select the best one to use to start the *Inter-Manager* negotiation. As stated before, in the first round, the list is ordered according to the manager utility function. On subsequent rounds, it is complemented by the q-learning algorithm and this process is explained in the next section.

7.6.3 Learning

As we detailed in Section 6.8 the GQN protocol has adaptive characteristics. These characteristics are related to the way the manager agents prepare a proposal formulation, using the Q-Learning algorithm (Watkins & Dayan, 1992) (in Section 7.6.3.7 we present an alternative implementation based on SARSA (Rummery & Niranjan, 1994)). There is another adaptive characteristic that we are including in the GQN protocol, related to the strategy used by the supervisor (or organizer) agent when starting new negotiations, i.e., we want the supervisor agent to learn from previous cases, solutions and feedback from the human-in-the-loop, and prepare the best strategy to start a negotiation when a similar problem appears. Because we did not perform experiments with this adaptive characteristics implemented, we give more details about it in the *Advanced Features* Section 7.8.

In MASDIMA the *manager* agents learn how to prepare new proposals after receiving feedback from the *supervisor* agent, through the use of Q-Learning. The goal is to react better and faster to the feedback received and, as such, prepare proposals that are closer to the preferences of the *supervisor* agent. As stated in Section 7.6.1 to prepare the proposal in the first round, the *manager* agents get a list of partial-solutions from the *specialist* agents and order them according to the utility function. With this ordered list, the *managers* start the *Inter-manager* negotiation with the goal of preparing the proposal. Instead of using this list on subsequent rounds, the *manager* agents request a new list of partial-solutions in every round. The reasons for this behavior are:

1. The environment is dynamic. Changes to flights, crew members, passengers, airport conditions, etc., might happen at any time.
2. It is necessary to use partial-solutions that comply with the feedback received. In MASDIMA we consider that a partial-solution that improves, at least, one of the attributes according to the feedback received from the *supervisor* is a compliant partial-solution.

Due to these reasons it is necessary to capture these changes in the partial-solutions before preparing new proposals. By using Q-Learning the *manager* agents are able to learn (adapt) its strategy in every round and not only at the end of the negotiation (Rocha & Oliveira, 1999) and, as such, taking into consideration the dynamics of the environment.

To use the Q-Learning algorithm in MASDIMA it is necessary to define what is a *State*, an *Action* and what *Reward Function* was used (Definitions 6.17,6.18 and 6.19, respectively) by each of the *manager* agents.

7.6.3.1 State

A *State* for the *A/C manager* was represented as

$$ST^{a/c} = \langle evc, ret, Class_{ad}, Class_{ac} \rangle \tag{7.19}$$

$Class_{ad}$ and $Class_{ac}$ is the feedback classification (i.e, *ok*, *high* or *low*) received from the *Supervisor* regarding the values of attribute *ad* and *ac*, respectively, presented in the previous proposal. Regarding the attributes from the *problem domain*, i.e., *evc* and *ret*, the former represents the *event that caused the problem*. From the study performed in Chapter 4 about the *Typical Problems* (Section 4.5) and the *Current Disruption Management Process* (Section 4.6) we identified the main causes and the possible values for the *evc* attribute. Table 7.6 shows this information.

Table 7.6 Possible values for the *evc* attribute

Values	Event Description
airp	Airport infrastructure and services causes. For example, unavailable parking stands, immigration issues, airport security problems, etc.
atc	En-route and destination Air Traffic Control restrictions and meteorological conditions.
comm	Protection of passengers in other flights due to the cancellation of a flight. Passengers missing after check-in.
crot	Crew member missing a flight due to the delay of a previous flight or other causes related to crew rotation.
hand	Events related to handling, e.g., boarding passengers and loading cargo.
induty	Crew member unavailable after sign-on and/or accident or illness during the flight or during the stay.
maint	Aircraft malfunctions and/or other events related to maintenance.
meteo	Meteorological conditions at the departure airport.
rot	Events related to aircraft rotation, e.g., late arrival of the incoming aircraft.
rules	Events related to exceeding any labor or law rules, like exceeding the daily duty time.
sec	Baggage identification, search and retrieve of baggage after boarding.
sign	Crew member not reporting for duty at home base.
other	Other events not included in the previous ones.

Regarding the latter attribute *ret* it represents the resource affected, in this case, the aircraft fleet. For example, A320 (Airbus 320) or A330 (Airbus 330).

Regarding the *Crew manager* the *State* was represented as

$$ST^{crew} = \langle evc, ret, Class_{cd}, Class_{cc} \rangle \tag{7.20}$$

As with the previous, $Class_{cd}$ and $Class_{cc}$ is the feedback classification received from the *Supervisor* regarding the values of attribute *cd* and *cc*, respectively. The values for the *evc* attribute are the same of Table 7.6 and, regarding the *ret*, instead of having aircraft fleets we have crew fleets, like *NB* (narrow body) and *WB* (wide body).

Finally, the *State* for the *Passenger manager* agent was represented as

$$ST^{pax} = \langle evc, ret, Class_{pt}, Class_{cc} \rangle \tag{7.21}$$

$Class_{pt}$ and $Class_{pc}$ is the feedback classification for the values of attribute *pt* and *pc*. The values for the *evc* attribute are the same as Table 7.6 and, regarding the *ret* the possible values are *BC* (Business Class) and *YC* (Economy Class).

7.6.3.2 Action

The *Action* for the *A/C manager* was represented as

$$AT^{a/c} = \langle pda, at_{ad}, at_{ac} \rangle$$

pda is the *problem domain action* for the aircraft dimension, i.e., the possible actions to use when solving the problems. From the study performed in Chapter 4 we identified the possible actions to use by the *specialist* agents of the *A/C manager* in Table 7.7.

Table 7.7 Possible values for the *A/C manager pda* attribute

Values	Description
cancel	Cancel the flight.
delay	Delay the flight a specific number of minutes.
exchange	Exchange the aircraft with the aircraft from another flight.
other	Lease crew and aircraft to another company (ACMI).

Regarding the attributes at_{ad} and at_{ac} they are the actions to be performed on the values of the attributes *ad* and *ac*, respectively, presented in the previous proposal. The possible values representing changes to be done on the next proposal's attributes are $\{increase, decrease, keep\}$. For example, if regarding the aircraft delay attribute (*ac*) the action $at_{ad} = increase$ this means that the manager should choose a partial-solution that increases the value of this attribute when compared to the solution presented in the previous proposal. At the end of this section we will explain better how the manager chooses the partial-solution that better corresponds to the action selected by the Q-Learning algorithm.

The *Crew manager Action* was represented as

$$AT^{crew} = \langle pda, at_{cd}, at_{cc} \rangle$$

The *problem domain actions* (*pda*) for the crew dimension are presented in Table 7.8 and like the ones for the *A/C manager* are the result of the study performed in Chapter 4. Regarding the attributes at_{cd} and at_{cc} the actions to be performed on the values of the attributes *cd* and *cc*, respectively, are the same for the *A/C manager*.

Finally, the *Passenger manager Action* was represented as

Table 7.8 Possible values for the *Crew manager pda* attribute

Values	Description
accept_delayed_crew	Use the delayed crew on the flight, implies to accept the delay.
exchange_crew	Exchange the delayed or missing crew with one from another flight.
other	Propose aircraft change and/or propose to cancel the flight.
proceed_without_crew	The flight will be performed without the missing or delayed crew member.
use_crew_on_vacation	Call a crew member that is on vacation to perform the flight.
use_dayoff_crew	Call a crew member that is on a day off to perform the flight.
use_free_time_crew	Use a crew member without any flight assigned.
use_reserve_crew	Use a crew member that is on reserve.

$$AT^{pax} = \langle pda, at_{pt}, at_{pc} \rangle$$

The *problem domain actions* (*pda*) for the passenger dimension are presented in Table 7.9. The actions represented by the attributes at_{pt} and at_{pc} are the same as the other managers'.

Table 7.9 Possible values for the *Passenger manager pda* attribute

Values	Description
change_flt_change_airl	Change the disrupted passenger to another flight from another airline company.
change_flt_same_airl	Change the disrupted passenger to another flight from the same company.
keep_same_flt	Keep the disrupted passenger on the disrupted flight.

It is important to point out that the *specialist agents* return the *action* (or *operator*) used to solve the problem and, as such, used to calculate the values of the attributes when sending the partial-solutions to the *managers*. Otherwise, it would not be possible to follow the action suggested by the learning algorithm. For example, a list l returned by a specialist agent of the aircraft dimension is represented as $l_{a/c} = \langle pda_1, ps_1, pda_2, ps_2, \cdots pda_n, ps_n \rangle$. More details about the specialist agents will be given in Section 7.7.

7.6.3.3 Reward Function and Example

To conclude the definition of the Q-Learning concepts we have used the following reward function for all *managers*:

$$Rw = \begin{cases} 2 & \text{, if winner} \\ \frac{2}{2} - \sum_i^2 pen_i & \text{, if looser.} \end{cases} \tag{7.22}$$

Since we want a faster convergence, we have defined the penalization according to the following ($0 \leq pen_i \leq 1$):

- If $Class_i = \{high\} \Rightarrow pen_i = 0.8$
- If $Class_i = \{low\} \Rightarrow pen_i = 0.2$
- If $Class_i = \{ok\} \Rightarrow pen_i = 0.0$

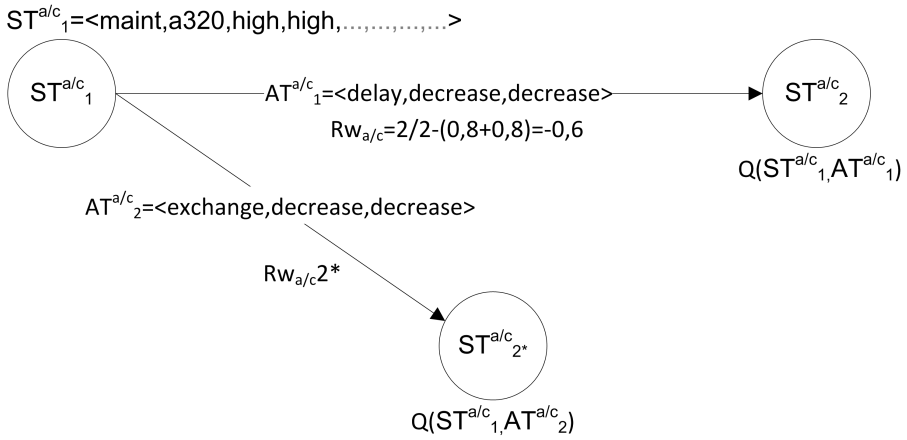


Fig. 7.9 Q-Learning in MASDIMA

Having defined the *State*, *Action* and *Reward Function* the manager agents are able to use the learning mechanism. To better understand how this works in MASDIMA, we are going to show a specific example. Figure 7.9 shows how the *A/C manager* agent used the learning mechanism to prepare a new proposal in a specific round t of the negotiation. In the previous round *t-1* the agent was in state $ST_1^{a/c}$ as the result of presenting a proposal with a possible solution $S_{t-1} =< 30,3000,5,1000,5,1500 >$ for which it received the feedback $F_{t-1} =< high, high, ok, ok, ok, ok >$. To present a proposal in round t the following will happen:

1. The action with the highest probability according to the Boltzmann formula (Equation 6.30) was selected. In this specific case it was $AT_1^{a/c} =< delay, decrease, decrease >$ (please note that each *manager* only cares about the feedback received for the dimension on which it has competence. In this example $< high, high >$).
2. Using action $AT_1^{a/c}$ as a reference the agent selected the partial-solution that is *nearer to the action*, i.e., one that has the same *problem domain action* (in this case *delay*) and that decreases the values presented in the previous solution. In this example it was selected the partial-solution $ps_1 =< 25,1500 >$. In the case of not having a partial-solution to choose from, then, a new action is selected according to step one above.
3. After engaging in an *Inter-Manager* negotiation, the agent presented a proposal with a possible solution $S_t =< 25,1500,10,700,10,1600 >$ for which it received the feedback $F_t =< high, low, ... >$.
4. The result of executing action $AT_1^{a/c}$ when it was in state $ST_1^{a/c}$ was the transition to state $ST_2^{a/c} =< maint, a320, high, low, ... >$, obtaining the reward $Rw_{a/c} = -0.6$ calculated according to Equation (7.22).
5. The Q-Value in state $ST_2^{a/c}$, i.e., $Q(ST_1^{a/c}, AT_1^{a/c})$ is updated according to Equation 6.31.

Regarding the above process, there are two things that we would like to better clarify. First, the way the agent chooses the partial-solution that is *nearer to the action* selected by the Q-learning (step 2 above) and, second, regarding the *default or initial Q-Values* used when a new pair $Q(ST, AT)$ is created (step 5 above). Regarding the latter we present more information in Sub-Section 7.6.3.6.

We have defined two strategies to select the partial-solution that is *near the action* selected by the Q-Learning. The first one, chooses the partial-solution whose attribute values are closer to the values presented on the previous proposal. This strategy is presented in Sub-Section 7.6.3.4. The

second one, chooses the partial-solution that gives the highest utility for the agent. It is presented in Sub-Section 7.6.3.5. It is important to point out that it is possible to defined many other strategies and that each agent can use a different strategy, if necessary.

7.6.3.4 Minimum Difference Strategy

Continuing with the above example, the action selected by the learning algorithm indicates that a partial-solution with delay as the *problem domain action* and that decreases both attributes should be the one choose from the list of possible partial-solutions. Remember that, from this list, the *manager* ignores the partial-solutions that do not comply with the feedback received, meaning that a partial-solution that does not improve the value of, at least, one of the attributes (when compared to the value of the previous proposal) is removed from the list. If the list has *only one* partial-solution compliant with the feedback, then the choice is done. If there is more than one partial-solution that decreases both attributes or, worst, that decreases one but increases the other, then the *manager* needs to make a choice. Lets call ps_1 to one of the solutions and ps_2 to the other with the following attributes values: $atr_1^{ps_1}$, $atr_2^{ps_1}$ and $atr_1^{ps_2}$, $atr_2^{ps_2}$. Let us also consider that the values of the attributes presented in the proposal of the previous round are atr_1^p, atr_2^p. The *manager* chooses the best partial-solution according to the following:

- Calculate the ratio of the attributes in relation to the previous values, i.e.

$$ratio_1^{ps_1} = atr_1^{ps_1}/atr_1^p$$

$$ratio_2^{ps_1} = atr_2^{ps_1}/atr_2^p$$

$$ratio_1^{ps_2} = atr_1^{ps_2}/atr_1^p$$

$$ratio_2^{ps_2} = atr_2^{ps_2}/atr_2^p$$

  If the ratio > 1 means that the attribute value increased, if it is < 1 means that the attribute value decreased and if $= 1$ means that it is equal.
- Calculate the absolute value of the distance of each attribute in relation to the previous values, i.e.,

$$dist_1^{ps_1} = \left|1 - ratio_1^{ps_1}\right|$$

$$dist_2^{ps_1} = \left|1 - ratio_2^{ps_1}\right|$$

$$dist_1^{ps_2} = \left|1 - ratio_1^{ps_2}\right|$$

$$dist_2^{ps_2} = \left|1 - ratio_2^{ps_2}\right|$$

- To the attributes that do not follow the direction suggested by the Q-Learning there is the application of a penalization *pen* greater than one and to the others equal to one.
- Evaluate each partial-solution using the Equation

$$eval = \sum_{i=1}^{2} (pen_i * dist_i)$$

- Choose the partial-solution that has the minimum *eval*.

Let us continue with the example of Figure 7.9 and use $S_{t-1} =< 30, 3000, 5, 1000, 5, 1500 >$ as the previous proposal and $AT_1^{a/c} =< delay, decrease, decrease >$ as the action suggested by Q-Learning. Suppose that the *A/C* manager has to choose from the following three possible partial-solutions: $ps_1 =< 20, 1500 >$, $ps_2 =< 25, 2500 >$ and $ps_3 =< 40, 1000 >$ the one that is *nearer to the action* suggested. Table 7.10 shows the result of applying the process above. The $ratio_1$ of ps_3 has a $pen_1 = 10$ because the value of this attribute has increased in relation to the previous one and the action suggested by the Q-learning was to decrease the value. The use of such an higher value results from experiments we have done, since we want to penalize heavily the partial-solutions that do not follow the suggestions given by the learning algorithm. As we can see, the partial-solution that is *nearer to the action* suggested by the Q-Learning algorithm is ps_2, then ps_1 and, finally, ps_3.

Table 7.10 Example of evaluation of partial-solution that are *near an action*

	$ratio_1$	pen_1	$ratio_2$	pen_2	$dist_1$	$dist_2$	**Eval**	**Order**
$AT_1^{a/c} =< decrease, decrease >$								
$S_{t-1} =< 30, 3000 >$								
$ps_1 =< 20, 1500 >$	0.66	1	0.50	1	0.34	0.50	0.84	2
$ps_2 =< 25, 2500 >$	0.83	1	0.83	1	0.17	0.17	0.34	1
$ps_3 =< 40, 1000 >$	1.33	10	0.33	1	0.33	0.67	7.03	3

7.6.3.5 Highest Utility Strategy

This strategy, when compared with the previous one, is very simple. From the list of partial-solutions that are compliant with the feedback received and with the action select by the learning algorithm, i.e., *problem domain action* = *delay* and that decreases both attributes, the following steps are performed:

- The utility of each partial-solution is calculated according to the interest of each agent.
- The list is ordered according to the utility value.
- The partial-solution with the highest utility value is selected.

7.6.3.6 Initial Q-Values

Regarding the *default or initial Q-Values* we could use zero as the initial value and, then, allow several iterations of the Q-Learning algorithm to happen so that it starts to converge to the best values. However and because we have collected information regarding the probability of applying specific actions to solve problems according to the event that originated those problems (see Tables 4.5, 4.6 and 4.7), we were able to speed up the convergence of the Q-Learning algorithm, by defining *initial Q-Values* according to the *evc* attribute of the *State* (the one that indicates the event cause for the problem) and the *pda* attribute of the *Action* (problem domain action suggested to be used to solve the problem). Table 7.11 shows these initial Q-Values that have a value between 0 and 1 defined according to the probability mentioned in Tables 4.5, 4.6 and 4.7.

Table 7.11: Initial Q-Values used in MASDIMA

Action	**Mng**	AIRP	ATC	COMM	CROT	HAND	INDUTY	MAINT	METEO	ROT	RULES	SEC	SIGN	OTH
cancel	a/c	0.05	0.05	0.1	0.05	0.02	0.05	0.02	0.02	0.01	0.05	0	0.05	0.03
delay	a/c	0.95	0.45	0.9	0.95	0.98	0.95	0.15	0.98	0.19	0.95	1	0.95	0.25
exchange	a/c	0	0	0	0	0	0	0.8	0	0.8	0	0	0	0.6
other	a/c	0	0.5	0	0	0	0	0.03	0	0	0	0	0	0.12
accept_delayed_crew	crew	0.27	0.27	0.27	0.6	0.27	0	0.27	0	0.27	0.25	0.27	0.5	0.27
exchange_crew	crew	0.06	0.06	0.06	0.1	0.06	0	0.06	0	0.06	0.1	0.06	0.1	0.06
other	crew	0.05	0.05	0.05	0	0.05	0.1	0.05	0.2	0.05	0.1	0.05	0.3	0.05
proceed_without_crew	crew	0.2	0.2	0.2	0.05	0.2	0.9	0.2	0	0.2	0.05	0.2	0.02	0.2
use_crew_on_vacation	crew	0.04	0.04	0.04	0.05	0.04	0	0.04	0	0.04	0.1	0.04	0.05	0.04
use_dayoff_crew	crew	0.06	0.06	0.06	0.1	0.06	0	0.06	0	0.06	0.1	0.06	0.1	0.06
use_free_time_crew	crew	0.05	0.05	0.05	0.05	0.05	0	0.05	0	0.05	0.1	0.05	0.1	0.05
use_reserve_crew	crew	0.27	0.27	0.27	0.6	0.27	0	0.27	0	0.27	0.25	0.27	0.5	0.27
change_flt_change_airl	pax	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
change_flt_same_airl	pax	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
keep_same_flt	pax	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2

7.6.3.7 SARSA

As we stated in the beginning of Section 7.6.3, we have also implemented an alternative reinforcement learning algorithm called SARSA (State-Action-Reward-State-Action) (Rummery & Niranjan, 1994). The goal is the same as the Q-Learning, i.e., to react better and faster to the feedback received and, as such, preparing proposals that are closer to the preferences of the *supervisor* agent.

The GQN protocol with SARSA works the same way as with the Q-Learning algorithm and, because of that, we will not repeat that information here. Although the goal of our work is not to compare the performance of these learning algorithms, with the implementation of SARSA we will be able to see how both algorithms perform when applied to the scenario of our experiments (Chapter 8).

As with Q-Learning, in SARSA there are the concepts of *state*, *action* and *reward* and we have defined them exactly the same for both algorithms.

Both algorithms are able to use the same policy regarding the selection of actions, e.g., *Softmax* or $\varepsilon - greedy$. The *Softmax* policy using the Boltzmann distribution is used in the implementation of the Q-Learning algorithm and is explained in Section 6.8.4. The $\varepsilon - greedy$ policy selects a random action with probability ε and the best action, i.e., the one that has the highest *Q-value* at the moment, with probability $1 - \varepsilon$. We have used $\varepsilon - greedy$ in the SARSA implementation.

The biggest difference between these two algorithms is the way they update the policy, i.e., the *Q(s,a)* value. SARSA uses the Equation (7.23) to update this value and follows an *on-policy* update, i.e., this value is updated according to the policy followed. Q-Learning uses the Equation (6.31) and follows an *off-policy* update, i.e., this value is updated based on the maximum reward of the available actions.

$$Q(st,at) = Q(st,at) + \alpha(rw + \gamma Q(st^*,a) - Q(st,at)) \tag{7.23}$$

7.7 Specialist Agents: The Problem Solving Experts

The architecture we have defined for MASDIMA supports a team of *specialist* agents for each manager. In the case of having more than one *specialist* agent in a team, each agent of that team should implement a different problem solving algorithm. Preliminary results show that a single problem solving algorithm is not able to solve, dynamically and within the required time restriction, all types of problems that we have identified during our observations (see Section 4.5). Taking advantage of the modularity, scalability and distributed characteristics of the MAS paradigm (see Section 7.2), we are able to add as many *specialist* agents as required, so that all types of problems are covered. Additionally, having the *specialist* agents of a team looking for solutions concurrently we are also able to take advantage of the concurrency and parallelism characteristics of the MAS contributing to a fault-tolerant system and, at the same time, speeding up the computation.

In this section we are going to present the specification of the *specialist* agents for the *aircraft*, *crew* and *passenger* teams. We have implemented two *specialist* agents for the *aircraft* team, two for the *crew* team and one for the *passenger* team. The *specialist* agents *AircraftHCSpecialist* and *CrewHCSpecialist* were implemented using the Hill Climb algorithm (Russell & Norvig, 2003) and the agents *AircraftSASpecialist* and *CrewSASpecialist* were implemented using the Simulated

Annealing algorithm (Kirkpatrick *et al.*, 1983). Regarding the *specialist* agent for the passenger team, *PassengerDijkstraSpecialist*, we have used the Dijkstra's algorithm (Dijkstra, 1959) because this problem can be easily modeled as a shortest path problem were the objective is to find the least costing path for each passenger to reach his destination airport from a starting airport.

The Hill Climb, Simulated Annealing and Dijkstra's algorithm are well known algorithms and we will not give details about them here. In Section 2.3 we explained how these algorithms work. In this section we will give in the first place, an informal description of the problem, goal and solution representation and, then a formal definition of the problem that each *specialist* agent should follow to implement a problem solving algorithm. This formalization is different for the *aircraft*, *crew* and *passenger* team.

7.7.1 Aircraft Specialist Agent

Informally, we can say that given a disrupted flight, the *aircraft specialist* agent should find a solution that guarantees the assignment of an aircraft to the disrupted flight, minimizing the flight delay and the flight costs (Equation (7.3)). A solution will be a set of pairs (**aircraft**, **flight**). The actions (or operators) available for the *aircraft specialist* agent in looking for a solution are the ones presented in Table 7.7.

Definition 7.1. *Flight Set and Disrupted Flight*
Let *Fl* represent the set of flights in a specific period of time, fl_j a specific flight of this set and *fd* a disrupted flight that also belongs to this set:

$$Fl = \{fl_1, fl_2, \ldots, fl_j\} \tag{7.24}$$

with

$$j \in N, 1 \leq j \leq |Fl|,$$

$$fl_j =< nbr_j, dairp_j, aairp_j, std_j, sta_j, tail_j, fleet_j, sseats_j, tseats_j, etd_j, eta_j >$$

$$fd =< nbr_{fd}, dairp_{fd}, aairp_{fd}, std_{fd}, sta_{fd}, tail_{fd}, fleet_{fd}, sseats_{fd}, tseats_{fd}, etd_{fd}, eta_{fd} >$$

where:

- *nbr* is the flight number;
- *dairp* is the departure airport;
- *aairp* is the arrival airport;
- *std* is the scheduled time of departure;
- *sta* is the scheduled time of arrival;
- *tail* is the tail number of the aircraft assigned;
- *fleet* is the aircraft fleet;
- *sseats* are the sold seats on the flight;
- *tsteats* are the total available seats in the flight;
- *etd* is the estimated time of departure;
- *eta* is the estimated time of arrival of the flight.

Definition 7.2. *Aircraft Set and Disrupted Aircraft*
Let *Ac* be the set of aircraft in a specific period of time, ac_j a specific aircraft of this set and *ad* the

disrupted aircraft of the disrupted flight *fd* that also belongs to this set:

$$Ac = \{ac_1, ac_2, \ldots, ac_j\} \tag{7.25}$$

with

$$j \in N, 1 \leq j \leq |Ac|,$$

$$ac_j =< tail_j, model_j, fleet_j, tseats_j, rtime_j >$$

$$ad =< tail_{ad}, model_{ad}, fleet_{ad}, tseats_{ad}, rtime_{ad} >$$

where:

- *model* is the aircraft model, e.g., A320, A330;
- *rtime* is the readiness time of the aircraft;
- *tail*, *fleet* and *tseats* have the same meaning as the ones in Equation (7.24).

Definition 7.3. *Aircraft Solution for a Disrupted Flight*
Let *Sac* be the set of aircraft and flight pairs that represent the solution for a specific disrupted flight and *sac* a specific pair of the solution:

$$Sac = \{sac_1, \ldots, sac_n\} \tag{7.26}$$

with

$$sac =< ac_i, fl_j >$$

$$n \in N, ac_i \in Ac, fl_j \in Fl$$

Definition 7.4. *Aircraft Solution Delay*
Let the Delay of an Aircraft Solution *Sac* be a function that calculates the total delay of a possible solution:

$$Sac \to \Re, Delay(Sac) := \sum_{1}^{n} \frac{(etd_n - std_n)}{\delta} \tag{7.27}$$

with

$$\delta \in \Re, 1 \leq n \leq |Sac|, \ \delta \ is \ \max(etd_n - std_n)$$

Definition 7.5. *Aircraft Solution Cost*
Let the Cost of an Aircraft Solution *Sac* be a function that calculates the aircraft cost of a possible solution according to Equation (7.3):

$$Sac \to \Re, Cost(Sac) := \sum_{1}^{n} \frac{accost(sac_n)}{\theta} \tag{7.28}$$

with

$$\theta \in \Re, 1 \leq n \leq |Sac|, \ \theta \ is \ \max(accost(sac_n))$$

Definition 7.6. *Aircraft Solution Penalization*
Let the Penalization of an Aircraft Solution *Sac* be a function that calculates the penalization that a possible solution has regarding changes that may difficult the other *manager agents* task:

$$Sac \to \Re, Penalization(Sac) := \sum_{1}^{n} (\gamma \times fleet_{pen} + \rho \times cap_{pen} + \mu \times pax_{pen}) \tag{7.29}$$

with

$$n \in N,\ 1 \leq n \leq |Sac|,$$

$$\gamma \in \Re,\ 0 \leq \gamma \leq 1,$$

$$\rho \in \Re,\ 0 \leq \rho \leq 1,$$

$$\mu \in \Re, 0 \leq \mu \leq 1,$$

$$fleet_{pen} = \begin{cases} 0 & \text{if } fleet_n = fleet_{fd} \\ 1 & \text{if } fleet_n \neq fleet_{fd} \end{cases}$$

$$cap_{pen} = \begin{cases} 0 & \text{if } sseats_n < tseats_{fd} \\ 1 & \text{if } sseats_n \geq tseats_{fd} \end{cases}$$

$$pax_{pen} = \begin{cases} 0 & \text{if } tseats_n \geq sseats_{fd} \\ 1 & \text{if } tseats_n < sseats_{fd} \end{cases}$$

The goal of $fleet_{pen}$ is to penalize solutions that use an aircraft that belongs to a different fleet than the disrupted aircraft. Using an aircraft from a different fleet might have an impact on the other agents solution. For example, it might require a different flight crew than the one in the disrupted flight, qualified for that type of aircraft.

Regarding the cap_{pen} the idea is to penalize the solutions that use aircraft assigned to flights that have more passengers than available seats on the disrupted flight, otherwise, we would have ($sseats_n - tseats_{fd}$) *disrupted passengers on another flight*, causing a new problem, and that will have in impact on the *passenger* manager.

Finally, regarding the pax_{pen} the idea is similar to the previous one: we want to penalize solutions that use aircraft assigned to flights that do not have enough available seats for the passengers of the disrupted flight. However, in this case and unlike the previous one, the consequence would be to have ($sseats_{fd} - tseats_n$) *disrupted passengers on the disrupted flight* having also an impact on the *passenger* manager.

Definition 7.7. *Aircraft Problem*
Considering the set of flights *Fl*, the set of aircraft *Ac* and given a disrupted flight *fd* and a disrupted aircraft *ad* we want to find the best solution *Sac*, minimizing the *Delay*, *Cost* and *Penalization*, i.e.:

$$min_{subject}(Delay(Sac) + Cost(Sac) + Penalization(Sac)) \tag{7.30}$$

$$\forall fl_k \in Fl,$$

$$fl_k =< nbr_k, dairp_k, aairp_k, std_k, sta_k, tail_k, fleet_k, sseats_k, tseats_k, etd_k, eta_k >,$$

$$\forall ac_n \in Ac,$$

$$ac_n =< tail_n, model_n, fleet_n, tseats_n, rtime_n >,$$

$$dairp_k = dairp_{fd}, \tag{7.31}$$

$$std_k > etd_{fd} \tag{7.32}$$

The expression (7.31) guarantees that only aircraft assigned to flights that are at the same airport as the airport of the disrupted flight are considered as possible solutions. Likewise, expression (7.32)

guarantees that only aircraft assigned to flights that have a scheduled time of departure after the estimated time of departure of the disrupted flight are considered as well.

7.7.2 Crew Specialist Agent

Given a *disrupted crew member* (Definition 3.3), the *crew specialist* agent should find a solution that guarantees the execution of the flight with the scheduled number of crew members or, in the worst case, with the minimum allowed crew members on board, minimizing the flight delay and the crew costs (Equation (7.4)). A solution will be a set of pairs (**crew**, **activity**). The actions (or operators) available for the *crew specialist* agent to look for a solution are the ones presented in Table 7.8.

Definition 7.8. *Activities Set and Disrupted Activity*
Let *As* represent the set of activities in a specific period of time, as_j a specific activity of this set and *asd* a disrupted activity that also belongs to this set:

$$As = \{as_1, as_2, \ldots, as_j\} \tag{7.33}$$

with

$$j \in N, 1 \leq j \leq |As|,$$

$$as_j =< type_j, sloc_j, sstart_j, send_j, estart_j, eend_j >$$

$$asd =< type_{asd}, sloc_{asd}, sstart_{asd}, send_{asd}, estart_{asd}, eend_{asd} >$$

where:

- *type* is the type of activity;
- *sloc* is the starting location;
- *sstart* is the scheduled starting time;
- *send* is the scheduled ending time;
- *estart* is the estimated starting time;
- *eend* is the estimated ending time.

Definition 7.9. *Crew Set and Disrupted Crew*
Let *Cw* be the set of crew members in a specific period of time, cw_j a specific crew member of this set and *cwd* a disrupted crew that also belongs to this set:

$$Cw = \{cw_1, cw_2, \ldots, cw_j\} \tag{7.34}$$

with

$$j \in N, 1 \leq j \leq |Cw|,$$

$$cw_j =< cnbr_j, rank_j, fleet_j, hbase_j, rtime_j >$$

$$cwd =< cnbr_{cwd}, rank_{cwd}, fleet_{cwd}, hbase_{cwd}, rtime_{cwd} >$$

where:

- *cnbr* is the crew member number;

- *rank* is the crew member rank;
- *fleet* is the crew member aircraft fleet;
- *hbase* is the crew member home base;
- *rtime* is the readiness time.

Definition 7.10. *Crew Solution for a Single Disrupted Crew Member*
Let *Scw* be the set of crew member and activity pairs that represent the solution for a specific disrupted crew member and *scw* a specific pair of the solution:

$$Scw = \{scw_1, \ldots, scw_n\} \tag{7.35}$$

with

$$scw =< cw_i, as_j >$$

$$n \in N, as_j \in As, cw_i \in Cw$$

Definition 7.11. *Crew Solution Delay*
Let the Delay of a Crew Solution *Scw* be a function that calculates the total delay of a possible solution:

$$Scw \to \Re, Delay(Scw) := \sum_{1}^{n} \frac{(estart_n - sstart_n)}{\delta} \tag{7.36}$$

with

$$\delta \in \Re, 1 \ \leq \ n \ \leq \ |Scw|, \ \delta \ is \ \max(estart_n - sstart_n)$$

Definition 7.12. *Crew Solution Cost*
Let the Cost of a Crew Solution *Scw* be a function that calculates the crew cost of a possible solution according to Equation (7.4):

$$Scw \to \Re, Cost(Scw) := \sum_{1}^{n} \frac{crewcost(scw_n)}{\theta} \tag{7.37}$$

with

$$\theta \in \Re, 1 \ \leq \ n \ \leq |Scw|, \ \theta \ is \ \max(crewcost(scw_n))$$

Definition 7.13. *Crew Solution Penalization*
Let the Penalization of a Crew Solution *Scw* be a function that calculates the penalization that a possible solution has regarding changes that may difficult the other *manager agents* task:

$$Scw \to \Re, Penalization(Scw) := \sum_{1}^{n} (\gamma \times hbase_{pen} + \rho \times poverl_{pen} + \mu \times toverl_{pen}) \tag{7.38}$$

with

$$n \in N, \ 1 \ \leq \ n \ \leq \ |Scw|,$$

$$\gamma \in \Re, \ 0 \leq \gamma \leq 1,$$

$$\rho \in \Re, \ 0 \leq \rho \leq 1,$$

$$\mu \in \Re, 0 \leq \mu \leq 1,$$

$$hbase_{pen} = \begin{cases} 0 & \text{if } hbase_n = hbase_{asd} \\ 1 & \text{if } hbase_n \neq hbase_{asd} \end{cases}$$

$$poverl_{pen} = \begin{cases} 0 & \text{if } (send_n \leq estart_{asd}) \vee (sstart_n \geq eend_{asd}) \\ 1 & \text{if } (sstart_n \leq estart_{asd} \leq send_n) \vee (sstart_n < eend_{asd}) \end{cases}$$

$$toverl_{pen} = \begin{cases} 0 & \text{if } (send_n \leq estart_{asd}) \vee (sstart_n \geq eend_{asd}) \\ 1 & \text{if } (sstart_n \geq estart_{asd}) \wedge (send_n \leq eend_{asd}) \end{cases}$$

$$poverl_{pen} \neq toverl_{pen} \tag{7.39}$$

The objective of $hbase_{pen}$ is to penalize solutions that use a crew member that has a *home base* different from the disrupted crew member. When using a crew member from another *home base*, besides the additional costs, the time necessary to position the crew member at the disrupted airport might have an impact on the other manager agents' solution.

With the $poverl_{pen}$ we want to penalize solutions that use crew members with an activity that overlays partially the disrupted activity. If that happens, although we might solve the problem of the disrupted crew member, we might also create a new problem and that might also have an impact on the other managers.

Regarding the $toverl_{pen}$ we want to penalize solutions that use crew members with an activity that overlays completely the disrupted activity. This situation is worst than the previous one.

Finally, the expression (7.39) means that we will not penalize twice the same solution regarding the activity overlay.

Definition 7.14. *Crew Problem*
Considering the set of activities *As*, the set of crew members *Cw* and given a disrupted activity *asd* and a disrupted crew member *cwd* we want to find the best solution *Scw*, minimizing the *Delay*, *Cost* and *Penalization*, i.e.:

$$min_{subject}(Delay(Scw) + Cost(Scw) + Penalization(Scw)) \tag{7.40}$$

$$\forall as_k \in As,$$

$$as_k =< type_k, sloc_k, sstart_k, send_k, estart_k, eend_k >,$$

$$\forall cw_n \in Cw,$$

$$cw_n =< cnbr_n, rank_n, fleet_n, hbase_n, rtime_n >,$$

$$sloc_k = sloc_{asd}, \tag{7.41}$$

$$sstart_k > estart_{asd} \tag{7.42}$$

$$DailyHours(cw_n) \leq DailyHoursLimit \tag{7.43}$$

$$WeeklyHours(cw_n) \leq WeeklyHoursLimit \tag{7.44}$$

$$MonthlyHours(cw_n) \leq MonthlyHoursLimit \tag{7.45}$$

The expression (7.41) guarantees that only crew members with activities starting at the same location as the disrupted activity, are considered as possible solutions. Likewise, expression (7.42) guarantees that only crew members with activities assigned that have a scheduled starting time after the estimated starting time of the disrupted activity are considered as well.

Finally, expressions (7.43), (7.44) and (7.45) will make sure that the crew members used as possible solutions will not exceed the *daily*, *weekly* and *monthly* work limits. Functions *DailyHours()*,

WeeklyHours() and *MonthlyHours()* calculate the work limits according to the rules defined for the company and country of the airline company.

7.7.3 Passenger Specialist Agent

Given a disrupted flight, the *passenger specialist* agent should find the best sequence of flights (itinerary) for each of the disrupted passengers, minimizing the passenger's trip time and the passenger's costs (Equations (7.5),(7.6)). A solution for all disrupted passengers will be a set of a sequence of flights from the disrupted airport to the final airport destination of each disrupted passenger.

As we stated in the beginning of this section we have used the Dijkstra's algorithm to implement the only *specialist* agent we have developed for the passenger team. Since this is a shortest-path algorithm, the possible solutions are not found by applying actions or operators like with the Hill Climb or Simulated Annealing algorithms. Nevertheless, if one wants to implement a *specialist* agent for this team that uses actions when looking for solutions, then the appropriate ones are presented in Table 7.9.

Definition 7.15. *Disrupted Passengers Set*
Let *Pxd* be the set of all disrupted passengers of the disrupted flight *fd* and pxd_i a specific passenger of this set:

$$Pxd = \{pxd_1, pxd_2, \ldots, pxd_i\} \tag{7.46}$$

with

$$i \in N, 1 \leq i \leq |Pxd|,$$

$$pxd_i =< id_i, dest_i, stt_i, sta_i, atd_i >$$

where:

- *id* is passenger identification;
- *dest* is the passenger final destination airport;
- *stt* is the scheduled trip time;
- *sta* is the scheduled time of arrival of the last flight of the passenger;
- *atd* is the actual time of departure of the first flight of the passenger;

Definition 7.16. *Passenger Solution for a Single Disrupted Passenger*
Let *Spx* be the ordered set of flights *fl* (according to Definition 7.1) that represent the solution for a specific disrupted passenger:

$$Spx = \{fl_1, \ldots, fl_n\} \tag{7.47}$$

with

$$n \in N, fl_n \in Fl$$

$fl_1 \in Spx$ is the first flight of the sequence and $fl_n \in Spx$ is the last flight.

Definition 7.17. *Passenger Solution Trip Time*
Let the Trip Time of a Passenger Solution *Spx* be a function that calculates the trip time of a possible solution:

$$Spx \rightarrow \Re, TripTime(Spx) \frac{(sta_n - atd_i)}{\delta} \tag{7.48}$$

with

$$\delta \in \Re,\ \delta\ is\ \max((sta_n - atd_i))$$

Definition 7.18. *Passenger Solution Cost*
Let the Cost of a Passenger Solution *Spx* be a function that calculates the passenger cost of a possible solution according to Equations (7.5) and (7.6):

$$Spx \rightarrow \Re, Cost(Spx) := \frac{(Pc(Spx) + Qc(Spx))}{\theta} \tag{7.49}$$

with

$$\theta \in \Re,\ \theta\ is\ \max(Pc(Spx) + Qc(Spx))$$

Definition 7.19. *Passenger Problem*
Considering the set of disrupted passengers *Pxd* and given a disrupted flight *fd* and a disrupted passenger *pxd* we want to find the best sequence of flights *Spx*, minimizing the *Trip Time* and *Cost*, i.e.:

$$min_{subject} \frac{(\gamma \times TripTime(Spx) + \omega \times Cost(Spx))}{(\gamma + \omega)} \tag{7.50}$$

$$\gamma \in \Re, 0 \leq \gamma \leq 1,$$

$$\omega \in \Re, 0 \leq \omega \leq 1$$

$$\forall fl_k \in Spx,$$

$$fl_k =< nbr_k, dairp_k, aairp_k, std_k, sta_k, tail_k, fleet_k, sseats_k, tseats_k, etd_k, eta_k >,$$

$$\forall pxd_n \in Pxd,$$

$$pxd_n =< id_n, dest_n, stt_n, sta_n, atd_n >,$$

$$\alpha \in \mathbf{N},$$

$$tseats_k > 0, \tag{7.51}$$

$$aairp_{k-1} = dairp_k, \tag{7.52}$$

$$std_k > sta_{k-1} + \alpha, \tag{7.53}$$

$$dairp_1 = dairp_{fd}, \tag{7.54}$$

$$aairp_n = dest_i, \tag{7.55}$$

The expression (7.51) guarantees that the flight has available seats. Expressions (7.52) and (7.53) make sure that the departure airport of the flight is the same as the arrival airport of the previous flight in the sequence and that the scheduled time of departure of the flight is after the scheduled time of arrival of the previous flight plus the *turn-around time* (i.e., the time necessary for an aircraft to be ready for departure after arriving from a previous flight).

The other two expressions, (7.54) and (7.55) have the objective to guaranteeing that the departure airport of the first flight in the sequence is the same as the disrupted airport and that the arrival airport of the last flight of the sequence is the same as the final destination of the disrupted passenger.

Definition 7.20. *Graph Representation*
Since we used a *graph* based algorithm, we have used for each pxd_i a direct weighted acyclic graph, i.e:

$$G = < Vrt, Edg > \tag{7.56}$$

where *Vrt* is the set of airports *Airp* that includes the disrupted airport of the passenger and the final destination airport, and *Edg* is the set of flights *Fl* that connects any two airports from the set *Vrt*.

7.8 Advanced Features

The purpose of this section is to point out features that, although mentioned in this chapter, were not conceptualized nor implemented and, as such, not used during the experimentation we have done. However, some of them have been studied and/or are being implemented as part of other projects or master thesis. First, we would like to remember the six features we have mentioned briefly in Section 7.3 and its advantages, i.e.:

1. *Maintenance Personal Tablet PCs*: Improves the communication between the *Maintenance Services* team and the maintenance personnel at the airport. Less errors when reporting events to the AOCC and better decision making is expected.
2. *Passengers Smartphones*: It will allow the passengers to receive alerts regarding disruptions, including new ETD and itineraries. It will also contribute to better decision making and, at the same time, improves the quality of service provided to the passengers.
3. *Crew members Tablet PCs*: Pilots and cabin crew can perform better, faster and with less errors their operational tasks. A better and wider integration of information can be achieved.
4. *Aircraft Datalink Connection*: Data exchange between the aircraft and ground information systems will allow a proactive action towards avoiding flight arrival delays. It will also contribute to better decision making.
5. *Alternative Transportation Datalink*: As it was proven by the 2010 Icelandic volcanic ash incident that caused flight disruptions which took days to be solved, under certain conditions, to have alternate itineraries performed by different means of transportation (e.g., bus, train) might be an acceptable solution. This is specially important to improve the disruption management process for the passenger part of the problem.
6. *Aircraft and Crew Electronic Market*: Nowadays, it is common for the airline companies to lease an aircraft and crew in some situations (called ACMI). The idea is to propose an electronic market that will allow to contract services regarding these two important resources.

There are two other main features that will improve the MASDIMA. At this moment we are already working on them but they were not finish on time to be included in this dissertation. These features are:

- *Human-in-the-loop*: As we show in Figures 7.1 and 7.2 the MASDIMA includes a *User Interface* agent that will allow a human user to have the final decision regarding the solution found by the system. Besides accepting or not the solution, the human user has the opportunity to provide feedback. Figure 7.10 shows a preliminary version of the *UI*. In the *Solution Plan* part of the UI the human can see the proposed solution and plan, i.e., can see the values for delay, costs and solution utility as well as the actions to be applied on the operational plan for each

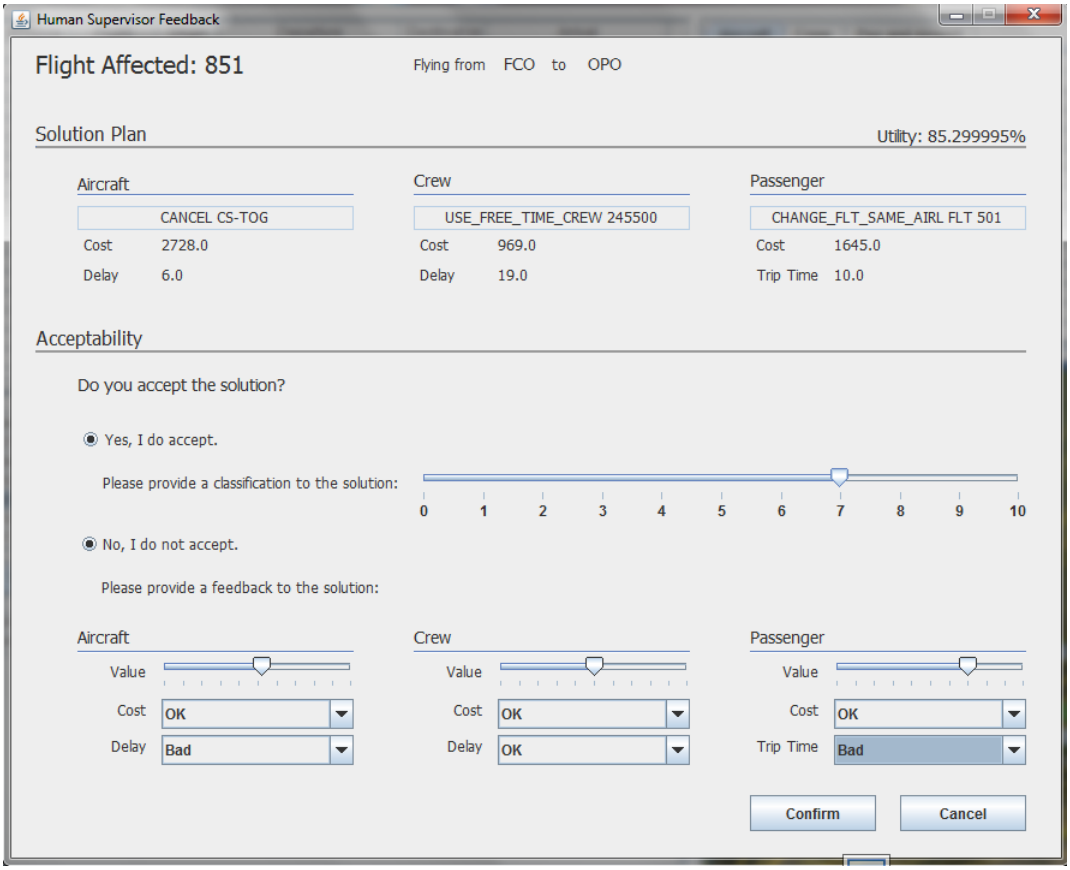


Fig. 7.10 Example of User Interface for *Human-in-the-loop* feature

dimension. In the *Acceptability* part of the *UI* it is possible to accept the solution and, in this case, classify quantitatively the solution. If the solution is not accepted the human user should provide quantitative and qualitative feedback regarding each of the dimensions and attributes, respectively. This way, the *human-in-the-loop* will be able to teach the system to react better not only in similar problems that might happen in the future but, also, in the specific problem that is being solved. For example, when the solution is not accepted, the *supervisor* agent will receive the feedback and it will react to that feedback by changing the range of acceptable values of the attributes and the weights of its utility function and start a new negotiation for the same problem by issuing a new *cfp*.

- *Learning with the past*: We want the *supervisor* agent to learn from previous cases, solutions and feedback from the *human-in-the-loop* and prepare the best strategy to start a new negotiation when a similar problem appears. With the current version of the MASDIMA the *supervisor* agent starts each new negotiation using default values for its preferences (e.g., range of acceptable values for the attributes and weights in the utility function). So, if a problem appears that is similar to the one that was solved in the past, the *supervisor* does not use that information. In this case and depending on the dynamics of the environment, it might reach the same solution taking the same amount of time it took previously. Our goal is to change this. Using the information from previous cases and the feedback, the *supervisor* agent can start the negotiation in a different way and, even, can pass information to the managers (as arguments or recommendations) that will lead the managers faster to the best solutions.

7.9 Chapter Summary

In this chapter we have presented our proposal for a *new approach for disruption management in the airline operations control domain*. It is a distributed, scalable and cooperative approach that encompasses not only the tools included, but also, the organization of the AOCC. The work presented here will help us to draw conclusions about the *first, third* and *fourth hypotheses* as formulated in Section 1.3.2.

The following sections describe our contributions to some of the characteristics mentioned above:

- In Section 7.4 about the *MASDIMA architecture*, we have introduced a functional, spatial and physical distribution approach. At the same time we took advantage of the scalable characteristic of the MAS as presented in Section 7.2.
- In Section 7.7 about *Specialist agents*, we improved the scalability of the system by allowing several agents to implement different algorithms that cover different types of problems. Additionally and since the *Specialist agents* look for candidate-solutions in every negotiation round, we are able to comply with two of requirements elicited in Section 4.8: To find solutions in real-time and to consider the dynamics of the environment.
- In Section 7.6 about the *Decision mechanisms*, we improved both the cooperation and adaptive capabilities of the system. Additionally, the decision mechanisms also contribute to comply with the requirements elicited in Section 4.8, specifically: (i) by allowing the different views that exist in the AOCC (*aircraft*, *crew* and *passenger*) to have the opportunity to be considered at the same level of importance when looking for solutions to the problems; (ii) by considering the local preferences of each team in the AOCC and (iii) by considering that the information may be incomplete and that the time may be limited.

We have also presented our vision, including organization and tools, that, in our opinion, will contribute to an *advanced and autonomous integrated AOCC*. As part of this future AOCC we propose some advanced features like the inclusion of alternative means of transportation (like bus and train) and an electronic market of aircraft and crew members, amongst others. These features were not implemented and, as such, are not part of the experiments we have performed to validate our approach. However, we believe that they have the potential to contribute in the future to a better disruption management process.

We have detailed the *MASDIMA architecture*, including how the system could be used in a *single* or *multiple-instance* architecture, allowing the system to be used in small, medium and large airlines.

Since utility functions are of the utmost importance for agent's decision-making process, we have done an exhaustive analysis of all costs that should be taken into account. As such, the *operational costs* involved, including *direct* and *quality* operational costs were explained. Regarding the quality operational costs we proposed a model that, in our opinion, captures best the relation that exists between the delay and the importance it has for the passengers in a flight. As part of that model we have proposed a way of determining the profiles of the passengers that might exist in a flight.

We show how we have applied the GQN Protocol (Chapter 6) as the decision mechanism and presented the two levels of negotiation used. In our opinion this negotiation approach is better then the current sequential approach adopted by most of the AOCC because it considers all the three dimensions at the same level of importance, leading to better integrated solutions.

The MASDIMA has adaptive characteristics, due to the use of Q-Learning by the *manager* agents, allowing them to learn how to formulate better proposals. In this chapter we have explained how we have applied the q-learning in this specific case.

One important component of our proposed system is the *Specialist agents*, i.e., the agents that implement *problem solving* algorithms, looking for possible solutions to the problems. We have one team of several specialist agents for each dimension of the problem in hands and the main idea is to include as many agents as necessary to be able to cover all types of problems, taking advantage of the scalable and concurrency characteristics of the MAS. In this chapter we have presented a detailed specification for the *aircraft*, *crew* and *passenger* specialist agents.

Finally, we have summarized the advanced features that, although not implemented, have been referred and included in some of the sections of this chapter.

In the next chapter we will present the *experiments* we have done with MASDIMA, including the scenarios, metrics and approaches used.

Chapter 8
Experiments

Abstract In Chapter 7 we have presented our new approach to disruption management in the airline domain. In this chapter we present the experiments we have performed using a system developed according to our proposal. This includes the scenario we have setup with the data and problems characterization, as well as the metrics we have defined related to *Air Transport*, *Negotiation Outcome*, *Solution Quality* and *Protocol Performance* which, we believe, are relevant for evaluating the system. We have also detailed the twelve approaches or methods used to solve the problems. The results as well as a critical discussion about them are presented.

8.1 Introduction

As we have stated in Chapter 1 we follow a *problem-oriented* line of research, applying the steps of the scientific method as specified in Section 1.3.

Step four of the methodology refers to the preparation of the *test laboratory*. In our case it corresponds to the development of the advanced prototype called *MASDIMA*. Chapters 6 and 7 provide conceptual information regarding the prototype and appendix A gives an overview of the implemented system.

A scientific method involves experiments to test if the proposed approaches, derived from the formulated hypotheses, adequately answer the questions and problems under investigation. Our methodology is not an exception and, in step five, we state that we have designed the experiments according to the hypotheses we have formulated in Section 1.3.2 and also that we have used the advanced prototype to perform those experiments.

This chapter is about the experiments we have designed and performed. In Section 8.1.1 we detail the scenario we have setup to perform the experiments related to disruption management in airline operations control. This includes the characterization of the data used and of the problems we want to solve, as well as, the agents and roles involved in the scenario. We believe that this is adequate to test the formulated hypotheses.

In Section 8.1.2 we identify and define the metrics used to compare the results obtained by the different approaches used to perform the experiments. We have included metrics related to *air transport*, *negotiation outcome*, *solution quality* and *protocol performance*.

In Section 8.1.3 we detail the approaches used to perform the experiments, i.e., *the different methods used to solve the problems*. We have included twelve approaches and variations of some of the approaches used.

In Section 8.2 we present the results of the experiments and analyze them using the chosen metrics. A discussion of the results is also presented.

Finally, we end this chapter with a summary in Section 8.3, highlighting the most important results obtained in this chapter.

8.1.1 Scenarios

In this section we detail the scenario we have setup to perform the experiments, including the characterization of the data used and the problems we want to solve. Ideally, to be able to fully test the hypotheses we have formulated in Section 1.3.2, we should use one year of real data from the airline operations plan of TAP Portugal. The air transport industry has seasonal behaviours (Tyler, 2012), making every month a little bit different regarding the occurrence of the events that cause disruptions.

Table 8.1 shows the percentage of TAP Portugal flight delays by event cause (see Table 4.2) from April 2009 to April 2010. As it is possible to see, regarding the event *METEO* (delays due to meteorological conditions at the departure airport), from October 2009 to March 2010 (Autumn and Winter months), there are much more flight delays due to bad weather than on the other months and this is only one example.

Table 8.1 Monthly percentage of delays for TAP Portugal from April 2009 to April 2010

Month	AIRP	ATC	COMM	HAND	MAINT	METEO	CREW	ROT	SEC	OTH
Apr/09	11	14	5	17	11	1	9	15	13	4
May/09	10	18	6	14	9	1	8	16	15	3
Jun/09	10	19	7	15	8	1	8	17	12	3
Jul/09	9	22	6	19	7	1	8	17	10	2
Aug/09	7	18	5	20	8	1	7	20	10	4
Sep/09	10	17	5	17	7	1	9	20	11	3
Oct/09	8	25	3	17	10	3	10	10	11	2
Nov/09	6	23	5	16	10	4	11	12	10	3
Dec/09	7	21	5	15	5	4	8	26	7	2
Jan/10	6	20	4	14	8	6	9	23	8	2
Feb/10	17	21	5	14	8	4	11	11	7	2
Mar/10	19	8	6	18	10	3	12	10	10	3
Apr/10	16	9	4	22	9	4	13	12	9	2
Average	**11**	**18**	**5**	**17**	**9**	**3**	**9**	**16**	**10**	**3**

Unfortunately, TAP Portugal's operational plan for one year corresponds to several TB (terabytes) of data, that need to be prepared for the data model used by our *test laboratory*, i.e., the MASDIMA system. Besides being a huge undertake to prepare such an amount of data, the experiments would also take a long time to be performed. Because of that we decided to use data for only one month of operation. We decided to use the data for September 2009 since it has *characteristics* similar to the average of one year of operation.

The September 2009 operational plan of TAP Portugal includes the information presented in Table 8.2. Besides the information related to the activity of the crew members and aircraft and flight schedule it also includes information related with operational costs, i.e.,

1. *Crew Costs* (salary, perdiem, hotel costs, etc.).
2. *Aircraft Costs* (airport, service and maintenance costs, fuel, etc.).
3. *Passenger Costs* (passenger compensations, loss of *goodwill*, etc.).

Details about these costs and how they were calculated are presented in Section 7.5 and in Appendix C.

Table 8.2 Information available in the operational plan

Table Name	Description
Activities	The crew members roster
Aircraft Models	Aircraft models average costs for ATC, Maintenance, Fuel and Handling
Aircrafts	The aircraft roster
Airport Charges	The charges applied by the airports
City Pairs	Latitude, Longitude and Distance between two airports
Crew Members	Group, Rank and Flight hours information for each crew member
Events	Events that caused problems on the operational plan (see Table 8.4)
Flights	Flight schedule from/to several origins and destinations
Hotel Charges	Hotel costs for passengers and crew members
Salaries	Salary information related to crew members

Table 8.3 characterizes the information presented in the operational plan for September 2009. As it is possible to see, it includes a very large number of flights (7931) as well as 55 aircraft, 3028 crew members and 585744 passengers. In our opinion, this information is representative and good enough to be used in our experiments.

Table 8.3 Operational Plan

Characteristic	Description
Total number of flights	7931
Departing from LIS	3251 (41,0%)
Departing from OPO	1026 (12,9%)
Departing from FNC	356 (4,5%)
Departing from FAO	105 (1,3%)
Departing from OTHER	3193 (40,3%)
Total passenger capacity	1091964 seats
Total seats sold	585744 (53,6%)
Total seats available	506230 (46,4%)
Total number of aircraft	39 Narrow Body (19 A319, 17 A320, 3 A321) 16 WB (12 A330, 4 A340)
Total number of crew members	3028 (783 flight crew, 2245 cabin crew)
By operational base	2909 in LIS, 105 in OPO and 14 in FNC
By crew rank:	
Captain (CPT)	414
First Officer (OPT)	369
Cabin Supervisor (SCB)	131
Purser (CCB)	473
Flight attendant (CAB)	1641

Out of the most representative events that happen in this month, we have randomly selected 49 events that caused the same number of problems in the operational plan. Table 8.4 characterizes the type of data included in table *Events* of the operational plan (Table 8.2). As it is possible to see, these events affected 49 flights, 31 aircraft, 286 crew members and 4760 passengers. Regarding the category of the events that caused the problems, and also in percentage, they are very similar to the average of one year of operation.

As we will explain in Section 8.1.3, the different approaches used to find the solutions to the problems are of two kinds: manual and automatic. Regarding the former, we have used the human operators of the AOCC with the organization, roles, functions and tools as explained in Section 4.3.

Table 8.4 Problem Data

Characteristic	Description
Flights affected	49
Aircraft Type affected	27 Narrow Body (14 A319, 10 A320, 3 A321) 4 Wide Body (3 A330, 1 A340)
Crew members affected	286
Number Passengers affected	576 business and 4184 Economy passengers
Total Expected Flight Delay	1752 mins. an average of 35,76 mins. per delayed flight
Total Schedule Aircraft Costs	93800 m.u. an average of 1914,29 m.u. per delayed flight
Total Schedule Crew Costs	98843 m.u. an average of 2017,20 m.u. per delayed flight
Event causes (see Tables 4.2,4.3)	6 *AIRP*, 9 *ATC*, 2 *COMM*, 1 *CROT* 8 *HAND*, 1 *INDUTY*, 5 *MAINT*, 2 *METEO* 1 *OTHER*, 1 *RULES*, 5 *SEC*, 1 *SIGN* 7 *ROT*

Regarding the latter, we have developed a MAS that represents the AOCC, including the agents, roles, functions and algorithms as explained in Section 7.4.

The information we use in this scenario is *static*, i.e., it does not change during the experiments we will perform. For example, there will not be new events affecting flights, besides the ones we have characterized here. However, in the real world, this is not the case. During the problem solving process, a new event might affect a resource that is being considered as a possible solution to the problem, making such solution infeasible. To try to capture this *dynamics* in the automatic approaches, we limit the number of candidate-solutions generated by each agent and, randomly, make sure that from one negotiation iteration to the next there is a subset of those candidate-solutions that is different. This is specially true and important in the case of the approaches that use automated negotiation with several rounds.

Finally, it is important to point out that the scenario, i.e., the operational plan and problems as characterized in this section, is the same for all approaches. This will allow to compare the different approaches and fairly compare the results obtained from each of them.

8.1.2 Metrics

Metrics or performance indicators are essential to evaluate the experimentation and compare the different approaches used (see Section 8.1.3). We have defined fourteen metrics related to the *Air Transport* domain, four related with *Solution Quality*, seven related with the *Negotiation Outcome* and five related with the *Protocol Performance*. In the next sub-sections and for each metric, we provide an identification, a definition, the process of calculating the metric and an interpretation. With this information, the reader will be able to better understand the results presented in Section 8.2. To better understand the metrics it is important to define the concepts of *experiment* and *experimental run*. The definition for these two concepts is as follows:

Definition 8.1. *Experiment*
An experiment corresponds to execute an *experimental run*, *n* times. We used *n=100*.

Definition 8.2. *Experimental Run*
An experimental run *e* consists of running a specific approach (see Section 8.1.3) one time for the 49 problems indicated and characterized in Section 8.1.1.

8.1.2.1 Air Transport Metrics

Definition 8.3. *Average Flight Departure Delay*
The average of *Flight Departure Delays* (in minutes) for an experiment of n experimental runs e is given by

$$\overline{FD} = \frac{\sum_{e=1}^{n}(\overline{FD}_e)}{n} \tag{8.1}$$

$\overline{FD}_e$ is the arithmetic average of the flight departure delay (FD) of all flights included in the experimental run e. A FD_i for a flight i is given by

$$FD_i = etd_i - std_i \tag{8.2}$$

where:
etd_i is the expected time of departure and std_i is the scheduled time of departure of flight i.
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to minimize the delays of the flights (on average).

Definition 8.4. *Average Crew Member Delay*
The average of *Crew Member Delays* (in minutes) for an experiment of n experimental runs e is given by

$$\overline{CwD} = \frac{\sum_{e=1}^{n}(\overline{CwD}_e)}{n} \tag{8.3}$$

$\overline{CwD}_e$ is the arithmetic average of the crew members delay (CwD) for all flights included in the experimental run e. A CwD_i for a flight i is given by

$$CwD_i = esign_i - ssign_i \tag{8.4}$$

where:
$esign_i$ is the expected *sign on* time and $ssign_i$ is the schedule *sign on* time of a crew member when reporting for duty i.
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to minimize the delays of crew members when reporting for duty (on average).

Definition 8.5. *Average Passenger Trip Time Delay*
The average of *Passenger Trip Time Delays* (in minutes) for an experiment of n experimental runs e is given by

$$\overline{PD} = \frac{\sum_{e=1}^{n}(\overline{PD}_e)}{n} \tag{8.5}$$

$\overline{PD}_e$ is the arithmetic average of the passenger trip time delay (PD) for all passengers of all flights included in the experimental run e. A PD_i for a passenger i is given by

$$PD_i = ett_i - stt_i \tag{8.6}$$

where:
ett_i is the expected trip time of the passenger i, i.e., the elapsed time between the actual time of departure of the first flight and the expected time of arrival of the last flight of the passenger itinerary, and

stt_i is the schedule trip time of the passenger i, i.e., the elapsed time between the schedule time of departure of the first flight and the schedule time of arrival of the last flight of the passenger itinerary (Definition 3.18).
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to minimize the delays of the passengers trip time (on average).

Definition 8.6. *Average Flight and Aircraft Costs*
The average of *flight and aircraft costs* (in m.u.) for an experiment of n experimental runs e is given by

$$\overline{FC} = \frac{\sum_{e=1}^{n}(\overline{FC}_e)}{n} \tag{8.7}$$

$\overline{FC}_e$ is the arithmetic average of the flight and aircraft costs (FC) of all flights included in the experimental run e. A FC_i for a solution i of a problem (a solution may include several flights) is given by Equation (7.3).
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to minimize the costs related to flight and aircraft. The several components included in this cost are explained in Section 7.5.1.1. It is important to point out that, *due to restrictions imposed by TAP Portugal* the absolute values of the costs have a difference (unknown for us) in relation to the real costs. However, for comparison purposes, the values are valid and we can draw conclusions from them.

Definition 8.7. *Average Crew Costs*
The average of *crew costs* (in m.u.) for an experiment of n experimental runs e is given by

$$\overline{CC} = \frac{\sum_{e=1}^{n}(\overline{CC}_e)}{n} \tag{8.8}$$

$\overline{CC}_e$ is the arithmetic average of the crew costs (CC) of all flights included in the experimental run e. A CC_i for a solution i of a problem (a solution may include several flights) is given by Equation (7.4).
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to minimize the costs related to crew members. The several components included in this cost are explained in Section 7.5.1.2.

Definition 8.8. *Average Passenger Costs*
The average of *passenger costs* (in m.u.) for an experiment of n experimental runs e is given by

$$\overline{PC} = \frac{\sum_{e=1}^{n}(\overline{PC}_e)}{n} \tag{8.9}$$

$\overline{PC}_e$ is the arithmetic average of the passenger costs (PC) for the *disrupted passengers* (see Definition 3.5) of all flights included in the experimental run e. A PC_i for a solution i of a problem (a solution may include several flights) is given by Equation (7.5).
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to minimize the costs related to passengers. The several components included in this cost are explained in Section 7.5.1.3.

Definition 8.9. *Average Flight Departure Delay Recovery Ratio*
The average of *Flight Departure Delay Recovery Ratio* for an experiment of n experimental runs

e is given by

$$\overline{FDrcv} = \frac{\sum_{e=1}^{n}(\overline{FDrcv}_e)}{n} \tag{8.10}$$

$\overline{FDrcv}_e$ is the arithmetic average of the flight departure delay recovery ratio ($FDrcv$) of all flights included in the experimental run e. A $FDrcv_i$ for a flight i is given by

$$FDrcv_i = \frac{FD_i}{OD_i} \tag{8.11}$$

where:
FD_i is the flight delay in minutes according to Equation (8.2) and OD_i is the original problem flight delay to be solved.
Interpretation: This ratio relates the minutes of the flight delays found by the approach used to perform experimental run e with the original flight delays in the problem used in the experimentation. A lower value is better. A value near zero means that the approach was able to recover the original flight delays. A value near one means that the approach did not recover well the flight delays or, if greater than one, means that it increased the original flight delays.

Definition 8.10. *Average Flight Cost Recovery Ratio*
The average of *Flight Cost Recovery Ratio* for an experiment of n experimental runs e is given by

$$\overline{FCrcv} = \frac{\sum_{e=1}^{n}(\overline{FCrcv}_e)}{n} \tag{8.12}$$

$\overline{FCrcv}_e$ is the arithmetic average of the flight cost recovery ratio ($FCrcv$) of all flights included in the experimental run e. A $FCrcv_i$ for a flight i is given by

$$FCrcv_i = \frac{FC_i}{OFC_i} \tag{8.13}$$

where:
FC_i is the flight cost in m.u. according to Equation (7.3) and OFC_i is the original problem flight cost to be solved.
Interpretation: This ratio relates the flight cost found by the approach used to perform experimental run e with the original flight cost in the problem used in the experimentation. A lower value is better, meaning that it was able to reduce the flight costs when compared with the original flight costs of the problem. A value equal to one means that the flight costs are the same and, if greater than one, that the flight costs are greater than the original ones.

Definition 8.11. *Average Crew Cost Recovery Ratio*
The average of *Crew Cost Recovery Ratio* for an experiment of n experimental runs e is given by

$$\overline{CCrcv} = \frac{\sum_{e=1}^{n}(\overline{CCrcv}_e)}{n} \tag{8.14}$$

$\overline{CCrcv}_e$ is the arithmetic average of the crew cost recovery ratio ($CCrcv$) of all flights included in the experimental run e. A $CCrcv_i$ for a flight i is given by

$$CCrcv_i = \frac{CC_i}{OCC_i} \tag{8.15}$$

where:
OC_i is the crew cost in m.u. according to Equation (7.4) and OCC_i is the original problem crew cost to be solved.
Interpretation: This ratio relates the crew cost found by the approach used to perform experimental run e with the original crew cost in the problem used in the experimentation. A lower value is better, meaning that it was able to reduce the crew costs when compared with the original crew costs of the problem. A value equal to one means that the crew costs are the same and, if greater than one, that the crew costs are greater than the original ones.

Definition 8.12. *Average Flight Cost per Minute*
The average of *Flight Cost per Minute* (in m.u. per minute) for an experiment of n experimental runs e is given by

$$\overline{FCmin} = \frac{\sum_{e=1}^{n}(\overline{FCmin}_e)}{n} \tag{8.16}$$

$\overline{FCmin}_e$ is the average of the flight cost per minute ($FCmin$) of all flights included in the experimental run e and is given by

$$FCmin_e = \frac{\overline{FC}_e}{\overline{OD}_e} \tag{8.17}$$

where:
$\overline{FC}_e$ is the arithmetic average of the flight and aircraft costs (FC) of all flights included in the experimental run e. A FC_i for a solution i of a problem (a solution may include several flights) is given by Equation (7.3) (see also Definition 8.6).
$\overline{OD}_e$ is the arithmetic average of the original problem flight delay to be solved included in the experimental run e.
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to produce, on average, a better flight and aircraft cost per minute for the original flight delays. It is useful to compare the different approaches because the original flight delays are the same on all approaches.

Definition 8.13. *Average Crew Cost per Minute*
The average of *Crew Cost per Minute* (in m.u. per minute) for an experiment of n experimental runs e is given by

$$\overline{CCmin} = \frac{\sum_{e=1}^{n}(\overline{CCmin}_e)}{n} \tag{8.18}$$

$\overline{CCmin}_e$ is the average of the flight cost per minute ($CCmin$) of all flights included in the experimental run e and is given by

$$CCmin_e = \frac{\overline{CC}_e}{\overline{OD}_e} \tag{8.19}$$

where:
$\overline{CC}_e$ is the arithmetic average of the crew costs (CC) of all flights included in the experimental run e. The CC_i for a solution i of a problem (a solution may include several flights) is given by Equation (7.4) (see also Definition 8.7).
$\overline{OD}_e$ is the arithmetic average of the original problem flight delay to be solved included in the experimental run e.
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to produce, on average, a better crew cost per minute for the original

flight delays. It is useful to compare the different approaches because the original flight delays are the same on all approaches.

Definition 8.14. *Average Passenger Cost per Minute*
The average of *Passenger Cost per Minute* (in m.u. per minute) for an experiment of n experimental runs e is given by

$$\overline{PCmin} = \frac{\sum_{e=1}^{n}(\overline{PCmin_e})}{n} \tag{8.20}$$

$\overline{PCmin_e}$ is the average of the passenger costs per minute ($PCmin$) for the *disrupted passengers* (see Definition 3.5) of all flights included in the experimental run e and is given by

$$PCmin_e = \frac{\overline{PC}_e}{\overline{OD}_e} \tag{8.21}$$

$\overline{PC}_e$ is the arithmetic average of the passenger costs (PC) for the *disrupted passengers* of all flights included in the experimental run e. A PC_i for a solution i of a problem (a solution may include several flights) is given by Equation (7.5) (see also Definition 8.8).
$\overline{OD}_e$ is the arithmetic average of the original problem flight delay to be solved included in the experimental run e.
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to produce, on average, a better passenger cost per minute for the original flight delays. It is useful to compare the different approaches because the original flight delays are the same on all approaches.

Definition 8.15. *Average Number of Flights with delays greater than 15 mins*
The average of the *number of flights with departure delays greater than 15 mins* for an experiment of n experimental runs e is given by

$$\overline{FD15min} = \frac{\sum_{e=1}^{n}(\overline{FD15min_e})}{n} \tag{8.22}$$

$\overline{FD15min_e}$ is the arithmetic average of the number of flights with departure delays greater than 15 minutes ($FD15min$) included in the experimental run e.
Interpretation: This metric provides the average number of flights with delays greater than 15 minutes. In the air transport industry, *punctuality* is usually calculated for flights delayed more than 15 minutes. The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to produce, on average, more punctual solutions.

Definition 8.16. *Percentage of Flights with delays greater than 15 mins*
The percentage of *flights with departure delays greater than 15 mins* for an experiment of n experimental runs e is given by

$$p\left(\overline{FD15min}\right) = \frac{\overline{FD15min_n})}{CP_e} \tag{8.23}$$

$\overline{FD15min_n}$ is the average number of flights with departure delays greater than 15 minutes according to Equation (8.22) and CP_e is the number of case problems included in the experimental run e.
Interpretation: This metric presents the *punctuality* in percentage and has the same goal of the metric $\overline{FD15min}$ defined in 8.15.

8.1.2.2 Negotiation Outcome Metrics

Definition 8.17. *Average Supervisor Agent Utility*
The average *supervisor agent utility* (a value in the range [0,1]) for an experiment of n experimental runs e is given by

$$\overline{Usup} = \frac{\sum_{e=1}^{n}(\overline{Usup}_e)}{n} \tag{8.24}$$

$\overline{Usup}_e$ is the arithmetic average of the negotiation winner's solution utility for the supervisor agent (*Usup*) for the case problems included in the experimental run e. The $Usup_i$ for a negotiation winner solution i of a problem is given by Equation (7.18).
Interpretation: The supervisor agent represents the AOCC integrated goals (i.e., to achieve solutions that include the three dimensions of the problem) and it corresponds to the organizer agent in the protocol terminology used in Chapter 6. This agent has a global view of the problem. The higher the utility of this agent the better, meaning that the approach used to perform experimental run e is producing solutions that correspond better to the interest of the agent.

Definition 8.18. *Average Aircraft Agent Utility*
The average *aircraft agent utility* (a value in the range [0,1]) for an experiment of n experimental runs e is given by

$$\overline{Ua/c} = \frac{\sum_{e=1}^{n}(\overline{Ua/c}_e)}{n} \tag{8.25}$$

$\overline{Ua/c}_e$ is the arithmetic average of the utility of the aircraft partial-solution included in the negotiation winner solution for the aircraft agent (*Ua/c*), for the case problems included in the experimental run e. The Ua/c_i for an aircraft partial-solution i of a problem is given by Equation (7.15).
Interpretation: The aircraft agent represents the flight and aircraft dimension of the problem in the AOCC. It is a respondent agent in the protocol terminology used in Chapter 6 and it has its own goals (different from the supervisor agent goals). This agent has a local view of the problem. The higher the utility of this agent the better for it. However, a lower utility does not mean a worst solution. Nevertheless, higher utilities for this agent and for the supervisor agent, are better.

Definition 8.19. *Average Crew Agent Utility*
The average *crew agent utility* (a value in the range [0,1]) for an experiment of n experimental runs e is given by

$$\overline{Ucrew} = \frac{\sum_{e=1}^{n}(\overline{Ucrew}_e)}{n} \tag{8.26}$$

$\overline{Ucrew}_e$ is the arithmetic average of the utility of the crew partial-solution included in the negotiation winner solution, for the crew agent (*Ucrew*), for the case problems included in the experimental run e. The $Ucrew_i$ for a crew partial-solution i of a problem is given by Equation (7.16).
Interpretation: The crew agent represents the crew dimension of the problem in the AOCC. It is a respondent agent in the protocol terminology used in Chapter 6 and it has its own goals (different from the supervisor and the other agent goals). This agent has its local views of the problems, like the aircraft's, and the interpretation of the utility values is similar.

Definition 8.20. *Average Passenger Agent Utility*
The average *passenger agent utility* (a value in the range [0,1]) for an experiment of n experimental runs e is given by

$$\overline{Upax} = \frac{\sum_{e=1}^{n}(\overline{Upax}_e)}{n} \tag{8.27}$$

$\overline{Upax}_e$ is the arithmetic average of the utility of the passenger partial-solution included in the negotiation winner solution, for the passenger agent (*Upax*), for the case problems included in the experimental run *e*. The $Upax_i$ for a passenger partial-solution *i* of a problem is given by Equation (7.17).
Interpretation: The passenger agent represents the passenger dimension of the problem in the AOCC. This agent also has a local view and the interpretation of this metric is similar to the aircraft's and crew's.

Definition 8.21. *Average Utilitarian Social Welfare*
The average *utilitarian social welfare* of the negotiation winner's solution for an experiment of *n* experimental runs *e* is given by

$$\overline{Usw} = \overline{Ua/c}_n + \overline{Ucrew}_n + \overline{Upax}_n \tag{8.28}$$

$\overline{Ua/c}_n$ is the average aircraft agent utility according to Equation (8.25). $\overline{Ucrew}_n$ is the average crew agent utility according to Equation (8.26) and $\overline{Upax}_n$ is the average passenger agent utility according to Equation (8.27).
Interpretation: This metric measures the utilitarian social welfare that, according to Sandholm (Sandholm, 1999), is "the global good that corresponds to the sum of the agents' individual utilities". It is important to point out that we are not claiming that the approaches used to perform the experiments produce a solution that maximizes social welfare. We use this metric to compare the global utility received by the agents that individually represent each dimension of the problem, i.e., aircraft, crew and passenger agent. Higher values are better, specially if combined with an higher average utility value for the supervisor agent ($\overline{Usup}$).

Definition 8.22. *Average Δ to the Domain Optimal Solution*
The average *Δ to the Domain Optimal Solution* (a value in the range [0,1]) for an experiment of *n* experimental runs *e* is given by

$$\overline{\Delta\,(optimal)} = \frac{\sum_{e=1}^{n}(\overline{\Delta\,(optimal)}_e)}{n} \tag{8.29}$$

$\overline{\Delta\,(optimal)}_e$ is the arithmetic average of the Δ to the domain optimal solution ($\overline{\Delta\,(optimal)}$) of all case problems included in the experimental run *e*. A $\overline{\Delta\,(optimal)}$ for a case problem *i* is given by

$$\overline{\Delta\,(optimal)}_i = Uopt_i - Usup_i \tag{8.30}$$

where:
$Uopt_i$ is the utility, for the supervisor agent, of the original schedule plan for the case problem *i*, i.e., before being disrupted. $Usup_i$ is the supervisor agent utility for the solution of case problem *i*, according to Equation (8.24).
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run *e* is able to produce solutions that are closer to the original schedule plan in terms of the utility for the supervisor agent. It is important to point out that we are not claiming that the approaches used to perform the experiments are producing *optimal solutions* from a mathematical point of view.

Definition 8.23. *Global Fairness*
The *global fairness* of the negotiation winner solutions for an experiment of *n* experimental runs *e*

is given by the *variance* of the average utility values for the agents that represent each dimension of the problem, i.e,

$$GF = var\left(\overline{Ua/c}_n; \overline{Ucrew}_n; \overline{Upax}_n\right) \tag{8.31}$$

$\overline{Ua/c}_n$ is the average aircraft agent utility according to Equation (8.25). $\overline{Ucrew}_n$ is the average crew agent utility according to Equation (8.26) and $\overline{Upax}_n$ is the average passenger agent utility according to Equation (8.27).
Interpretation: One of the expected results of this thesis is to *balance the utility of each agent that represents a dimension of the problem, contributing to a more equal importance of each dimension without implying a decrease in the supervisor agent utility* (See Section 1.3.3). The goal of this metric is to measure this *balance* through the *variance* of the agent's utility values. Values near zero are better.

8.1.2.3 Solution Quality Metrics

Definition 8.24. *Aircraft Solution Quality*
The *aircraft solution quality* (a value in the range [0,1]) for an experiment of n experimental runs e is given by

$$A/Cqual = \frac{\frac{\sum_{i=1}^{a}(\left|(\overline{A/Cact_n})_i - (A/Cact_p)_i\right|)}{100}}{a} \tag{8.32}$$

a is the number of *aircraft actions* that are possible to be used for solving the aircraft part of the problem (see *interpretation* below).
$(A/Cact_p)_i$ is the preferred value for the percentage of times the *aircraft action i* should be used for solving the aircraft part of the problem.
$\overline{A/Cact_{n_i}}$ is the arithmetic average of the percentage of times that the *aircraft action i* was used in an experiment of n experimental runs e and is given by

$$\overline{A/Cact_{n_i}} = \frac{\overline{A/Cact_{e_i}}}{n} \tag{8.33}$$

$\overline{A/Cact_{e_i}}$ is the arithmetic average of the percentage of times that the *aircraft action i* was used in the experimental run e and is given by

$$\overline{A/Cact_{e_i}} = \frac{(A/Cact_e)_i}{CP_e} \tag{8.34}$$

where:
$(A/Cact_e)_i$ is the number of times that action i was used in the aircraft solution for all problems included in experimental run e.
CP_e is the number of case problems included in the experimental run e.
Interpretation: From the interviews performed to the human controllers at TAP Portugal's AOCC (Section 4.6) several actions used to solve the aircraft part of the problem were identified, as well as the probability of using them according to the event cause. This metric measures the quality of the aircraft solution found by the approach used to perform the experimental run e, according to this information. It is important to point out that all solutions found by each of the approaches are valid. As such, the intention of this metric is to measure the solution (or part of it) according to the

preferences of the AOCC users, regarding the type of actions used to solve the problems. These preferences result from the knowledge and experience that the users have in solving the problems. The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to produce solutions (or part of it) using actions that are closer to the preferences of the AOCC users.

Definition 8.25. *Crew Solution Quality*
The *crew solution quality* (a value in the range [0,1]) for an experiment of n experimental runs e is given by

$$CRWqual = \frac{\frac{\sum_{i=1}^{a}(\left|(\overline{CRWact_n})_i - (CRWact_p)_i\right|)}{100}}{a} \tag{8.35}$$

a is the number of *crew actions* that are possible to be used for solving the crew part of the problem (see *interpretation* below).
$(CRWact_p)_i$ is the preferred value for the percentage of times the *crew action i* should be used for solving the crew part of the problem.
$\overline{CRWact_{ni}}$ is the arithmetic average of the percentage of times that crew action i was used in an experiment of n experimental runs e and is given by

$$\overline{CRWact_{ni}} = \frac{\overline{CRWact_{ei}}}{n} \tag{8.36}$$

$\overline{CRWact_{ei}}$ is the arithmetic average of the percentage of times that crew action i was used in the experimental run e and is given by

$$\overline{CRWact_{ei}} = \frac{(CRWact_e)_i}{CP_e} \tag{8.37}$$

where:
$(CRWact_e)_i$ is the number of times that action i was used in the crew solution for all problems included in experimental run e.
CP_e is the number of case problems included in the experimental run e.
Interpretation: This metric has a similar interpretation to the *Aircraft Solution Quality* metric. Please see Definition 8.24 for more information. The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to produce solutions (or part of it) using actions that are closer to the preferences of the AOCC users.

Definition 8.26. *Passenger Solution Quality*
The *passenger solution quality* (a value in the range [0,1]) for an experiment of n experimental runs e is given by

$$PAXqual = \frac{\frac{\sum_{i=1}^{a}(\left|(\overline{PAXact_n})_i - (PAXact_p)_i\right|)}{100}}{a} \tag{8.38}$$

a is the number of *passenger actions* that are possible to be used for solving the passenger part of the problem (see *interpretation* below).
$(PAXact_p)_i$ is the preferred value for the percentage of times the *passenger action i* should be used for solving the passenger part of the problem.
$\overline{PAXact_{ni}}$ is the arithmetic average of the percentage of times that passenger action i was used in an experiment of n experimental runs e and is given by

$$\overline{PAXact_{ni}} = \frac{\overline{PAXact_{ei}}}{n} \tag{8.39}$$

$\overline{PAXact_{ei}}$ is the arithmetic average of the percentage of times that passenger action i was used in the experimental run e and is given by

$$\overline{PAXact_{ei}} = \frac{(PAXact_e)_i}{CP_e} \tag{8.40}$$

where:
$(PAXact_e)_i$ is the number of times that action i was used in the passenger solution for all problems included in experimental run e.
CP_e is the number of case problems included in the experimental run e.
Interpretation: This metric has a similar interpretation to the *Aircraft Solution Quality* metric. Please see Definition 8.24 for more information. The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to produce solutions (or part of it) using actions that are closer to the preferences of the AOCC users.

Definition 8.27. *Integrated Solution Quality*
The *integrated solution quality* (a value in the range [0,1]) for an experiment of n experimental runs e is given by

$$ITGqual = \frac{A/Cqual_n + CRWqual_n + PAXqual_n}{3} \tag{8.41}$$

$A/Cqual_n$ is the aircraft solution quality according to Equation (8.32).
$CRWqual_n$ is the crew solution quality according to Equation (8.35).
$PAXqual_n$ is the passenger solution quality according to Equation (8.38).
Interpretation: The goal of this metric is to measure the quality of the integrated solution (that which includes the three dimensions of the problem) according to the AOCC users preferences. A lower value is better. Please see the *interpretation* section of the metric *Aircraft Solution Quality* metric (Definition 8.24) for more information.

8.1.2.4 Protocol Performance Metrics

Definition 8.28. *Average Number of Rounds to Reach an Agreement*
The average *number of rounds to reach an agreement* for an experiment of n experimental runs e is given by

$$\overline{NR} = \frac{\sum_{e=1}^{n}(\overline{NR}_e)}{n} \tag{8.42}$$

$\overline{NR}_e$ is the arithmetic average of the number of rounds needed to reach an agreement ($\overline{NR}$) of all case problems included in the experimental run e.
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to reach an agreement in fewer negotiation rounds.

Definition 8.29. *Average Negotiation Time*
The average *negotiation time* for an experiment of n experimental runs e is given by

$$\overline{NT} = \frac{\sum_{e=1}^{n}(\overline{NT}_e)}{n} \tag{8.43}$$

$\overline{NT}_e$ is the arithmetic average of the negotiation time ($\overline{NT}$) of all case problems included in the experimental run e. The negotiation time NT_i for a case problem i is the elapsed time measured from the first *call-for-proposal* issued by the supervisor agent to the information of the negotiation winner agent also provided by the supervisor agent.
Interpretation: A lower value in this metric is better. However, this value is highly dependent on the computer and physical distribution of the multi-agent system (MAS) as well as on the algorithms used to find the candidate-solutions to the problems. Additionally, due to the functional distribution (e.g., having agents working in parallel) and when compared with sequential approaches, the MAS takes less time.

Definition 8.30. *Average Search Time*
The average *search time* for an experiment of n experimental runs e is given by

$$\overline{ST} = \frac{\sum_{e=1}^{n}(\overline{ST}_e)}{n} \tag{8.44}$$

$\overline{ST}_e$ is the arithmetic average of the search time ($\overline{ST}$) of all case problems included in the experimental run e. The search time ST_i for a case problem i is the elapsed time measured from the first *call-for-proposal* issued by the manager agents (aircraft, crew and passenger) to the specialist agents (the ones that implement a problem solving algorithm) until the moment the manager agents have a proposal ready to present to the supervisor.
Interpretation: A lower value in this metric is better. However, this metric has the same dependencies as the ones stated in the *negotiation time* metric (Definition 8.29).

Definition 8.31. *Average Number of Messages Exchanged*
The average *number of messages exchanged* for an experiment of n experimental runs e is given by

$$\overline{Msg} = \frac{\sum_{e=1}^{n}(Msg_e)}{n} \tag{8.45}$$

Msg_e is the number of messages exchanged by all agents (Msg) during the negotiation for all case problems included in the experimental run e.
Interpretation: The lowest this value is, the better, meaning that the agents were able to reach an agreement by exchanging less messages.

Definition 8.32. *Average Number of Messages per Problem*
The average *number of messages per problem* exchanged by the agents for an experiment of n experimental runs e is given by

$$\overline{Mprb} = \frac{\sum_{e=1}^{n}(Mprb_e)}{n} \tag{8.46}$$

$Mprb_e$ is the number of messages per problem ($Mprb$) for all case problems included in the experimental run e and is given by

$$Mprb_e = \frac{Msg_e}{\overline{CP}_e} \tag{8.47}$$

where:
Msg_e is the number of messages exchanged by all agents during the negotiation for all case problems included in the experimental run e.
CP_e is the number of case problems included in the experimental run e.
Interpretation: The lowest this value is, the better, meaning that the approach used to perform experimental run e is able to generate less messages per problem when reaching an agreement.

8.1.3 Approaches

We define approaches as the *different methods used to solve the problems* in the scenario presented in Section 8.1.1. We have used one *manual* (human based) approach and several *automatic* ones.

Regarding the manual approach, called ***TAP-AOC***, we have used the *historical data from TAP Portugal*, regarding the way the human operators of the AOCC solved the problems. The organization, roles, functions, tools and the process used are the ones described in Sections 4.3, 4.4 and 4.6. Being a manual process, several of the metrics we defined in Section 8.1.2 are not usable here. For example, the ones related to the *negotiation outcome*. Even regarding the *air transport* metrics, we are not able to use all the defined metrics because the information available is not complete. Nevertheless, and since the scenario regarding the data available is the same for the manual and automatic approaches, we decided to include this approach because it will allow to draw conclusions regarding the use of a manual approach versus an automatic one.

Regarding the automatic approaches, besides sharing the same scenario with the manual one, they are based on the MAS paradigm and they include the agents and roles as specified in Section 7.4 as well as the problem solving agents as specified in Section 7.7. We implemented eleven automatic approaches and ten of them use the decision mechanism specified in Section 7.6 based on automatic negotiation. The one that does not use automatic negotiation is called ***TSA - Traditional Sequential Approach*** and uses a sequential approach.

As we have described in Section 3.4 (*Airline Operations Recovery*) and as we have explained in Chapter 4 (*The Airline Operations Control Problem*), the majority of the current work of other researchers as well as the current AOCC disruption management process, typically follow a sequential approach when solving a disruption, i.e., the *aircraft* dimension of the problem is solved first, then, using the aircraft partial-solution as the input, the *crew* dimension is solved (in the case of not having a crew available, another aircraft partial-solution is retrieved) and, finally, using the aircraft and crew parts of the solution as the input, a solution for the *passenger* dimension is found. The *TSA* approach implements this method. Since *TSA* is an automatic approach we are able to collect more information and, as such, more metrics can be used.

The other ten automatic approaches use the *GQN - Generic Q-Negotiation* protocol as the decision mechanism. The conceptual definition of the protocol is presented in Chapter 6 and the application of the protocol to the disruption management problem in AOCC is presented in Section 7.6. The differences between these ten approaches will be explained later on in this section.

For now, it is important to define the several parameters (preferred and maximum values) and weights used by the evaluation formulas and utilities of the several agents included in the automatic approaches[1]. Each of the agents *supervisor*, *aircraft manager*, *crew manager* and *passenger manager* have their own evaluation formulas and utility functions. Equation 7.18 is the one used by the supervisor agent and Equations 7.15, 7.16 and 7.17 are the ones used by the aircraft, crew and passenger manager, respectively. Table 8.5 shows the weights, preferred values (for supervisor only) and maximum values used in the utility function for each agent. All automatic approaches share the same utility functions, weights and parameter values.

It is important to point out that the values presented here for the weights and parameters, were selected and validated by the *human operators* of TAP Portugal's AOCC. We have performed

[1] Although the *TSA* approach does not use negotiation we calculate the utilities for each agent, since in this approach they are used as a way of ordering the candidate-solutions. Additionally, it will allow a better comparison with the other approaches.

Table 8.5 Weights and parameters for the agents utility functions (Equations 7.18, 7.15, 7.16 and 7.17)

Agent	Values
Supervisor Agent	
Weights	$\alpha_1 = 0,34; \alpha_{2,3} = 0,33; v_1 = 0,43; v_2 = 0,14; v_3 = 0,28$
	$v_4 = 0,05; v_5 = 0,05; v_6 = 0,05$
Preferred Values	A/C and Crew Delay: 0
	Passenger Trip Time Delay: 0
	A/C and Crew Cost: 1,05 X Schedule Cost
	Passenger Cost: 10 X Total Pax
Maximum Values	A/C, Crew and Passenger delays: 120
	A/C and Crew Cost: 1,5 X Schedule Cost
	Passenger Cost: 100 X Total Pax
Aircraft Manager Agent	
Weights	$w_1 = 1; w_2 = 0,11$
Maximum Values	A/C, Crew and Passenger delays: 120
	A/C Cost: 4400; Crew Cost: 3500; Passenger Cost: 9800
Crew Manager Agent	
Weights	$w_3 = 0,33; w_4 = 0,11$
Maximum Values	A/C, Crew and Passenger delays: 120
	A/C Cost: 4400; Crew Cost: 3500; Passenger Cost: 9800
Passenger Manager Agent	
Weights	$w_5 = 0,66; w_6 = 0,11$
Maximum Values	A/C, Crew and Passenger delays: 120
	A/C Cost: 4400; Crew Cost: 3500; Passenger Cost: 9800

several experiments, with different values, and these were the ones that generated more well-balanced and acceptable solutions according to their opinion.

Besides testing the hypotheses, the general idea of performing experiments with this number of approaches, is to test different strategies used by the agents when presenting proposals and to see the impact of having a dynamic environment where the number of candidate-solutions can change unexpectedly.

Out of those ten approaches that are based on *GQN*, one does not use any learning mechanism, five use the *Q-Learning* mechanism as explained in Section 7.6.3 and four use the *SARSA* mechanism as explained in Section 7.6.3.7. A description of each one as well as their mutual differences follows.

- ***FB10*** *(Feedback only, maximum ten candidate-solutions)*: This approach does not use any learning mechanism. The *specialist* agents (the ones that generate candidate-solutions, see Section 7.7) generate candidate-solutions with domain actions (e.g., exchange an aircraft, use a reserve crew, etc.) according to the *probabilistic solution action* (see Section 4.6) obtained from interviewing the human operators in TAP's AOCC. The supervisor gives feedback and the manager agents (aircraft, crew and passenger) choose from the candidate-solutions the one that complies with the feedback and that has the highest utility. It is with this candidate-solution that the *inter-manager* negotiation is initiated. A maximum of ten candidate-solutions will be sent to each of the managers and, from one round to another, some of the candidate-solutions are not repeated

so that the *dynamics* of the information is considered (see Section 8.1.1). The negotiation was limited to ten rounds.

- ***Q10-Min*** *(Q-Learning, maximum ten candidate-solutions, Minimum Difference Strategy)*: This approach uses a learning mechanism based on the Q-Learning algorithm (see Section 7.6.3). The learning mechanism is used by the managers in the negotiation with the supervisor and, also, in the *inter-managers* negotiation. The Q-Learning algorithm starts with the *default Initial Q-Values* as explained in Section 7.6.3.6. The *specialist* agents generate candidate-solutions without considering the *probabilistic solution action* mentioned in the previous approach, i.e., all the domain actions have the same probability to be used. The supervisor agent gives feedback to the proposal presented by the managers. The managers, of the candidate-solutions presented by the specialists, do not consider the ones that do not follow the action suggested by the Q-Learning algorithm. Out of the remaining, the managers select the one that is *nearest to the previous proposal* according to the strategy defined in Section 7.6.3.4. It is with this candidate-solution that the negotiation *inter-manager* is initiated. Like in the previous approach, a maximum of ten candidate-solutions will be sent to each of the managers and, from one round to another, some of the candidate-solutions are not repeated so that the *dynamics* of the information is considered (see Section 8.1.1). The negotiation was limited to ten rounds.
- ***Q10-Best*** *(Q-Learning, maximum ten candidate-solutions, Best Utility Strategy)*: This approach is very similar to the *Q10-Min*. The only difference is that the manager agents follow the *Best Utility* strategy when selecting the candidate-solution from the set of candidate-solutions that follow the suggested action presented by the Q-Learning algorithm (see Section 7.6.3.5.
- ***Q10-Best-Filter*** *(Q-Learning, maximum ten candidate-solutions, Best Utility Strategy, Filtered by the domain operator)*: This approach is very similar to the *Q10-Best*. The only difference is that the managers discard the candidate-solutions presented by the specialists that do not follow the action suggested by the Q-Learning as well as the ones that do not follow the *domain operator* also suggested by the learning mechanism. This *small* difference reduces the number of candidate-solutions available to be chosen by the managers.
- ***Q20-Best*** *(Q-Learning, maximum twenty candidate-solutions, Best Utility Strategy)*: This approach is similar to the *Q10-Best* but, instead of having a maximum of ten candidate-solutions per manager it has a maximum of twenty candidate-solutions. This means that the managers have more candidate-solutions to choose from, increasing the possibility of getting higher utilities.
- ***S10-Best*** *(SARSA, maximum ten candidate-solutions, Best Utility Strategy)*: The only difference between this approach and the *Q10-Best* is that it uses the SARSA learning algorithm instead the Q-Learning one (see Section 7.6.3.7).
- ***S10-Best-Filter*** *(SARSA, maximum ten candidate-solutions, Best Utility Strategy, Filtered by the domain operator)*: The same as the *Q10-Best-Filter* approach but using the SARSA learning algorithm.
- ***S20-Best*** *(SARSA, maximum twenty candidate-solutions, Best Utility Strategy)*: The same as the *Q20-Best* approach but using the SARSA learning algorithm.
- ***Q10-Best-V2*** *(Q-Learning, maximum ten candidate-solutions, Best Utility Strategy, Improved version)*: This approach and the next one, have some differences among them regarding the way the learning mechanism is used. However, everything else is similar to the approach *Q10-Best*. The main idea of this approach is to see the impact of not using learning mechanisms during the inter-manager negotiation. The differences are:

- A learning mechanism *is not used* in the inter-managers negotiation, i.e., the manager agents when replying to a request from the manager that initiated the inter-managers negotiation, will comply with the restrictions but, instead of choosing the best partial candidate-solution based on the action suggested by the learning mechanism, will select the one with the highest utility for itself. This also increases the number of available partial candidate-solutions to choose from.
- A new element *very_high* was added to the n-tuple F (see Definition 6.5).
- The reward formula Rw (Equation 6.36) was updated to include the six attributes of the proposal (m=6) and not only the two that corresponds to the dimension of the manager agent. The penalty for $Class_i = \{very\_high\}$ was defined as $pen_i = 1,5$. The penalty for the attributes of the other dimensions has a weight of 0,8 instead of 1 as the ones that belong to the dimension of the manager.
- The state definition for the aircraft, crew and passenger manager agent (Definitions 7.19, 7.20 and 7.21, respectively) include a new attribute $Class_{oth} = \{good, bad, very\_bad\}$ that is defined according to the classification obtained for the other attributes that do not belong to the dimension of the manager.

- ***Q20-Best-V2***: This approach is similar to the *Q10-Best-V2* but instead of being generated a maximum of ten candidate-solutions per manager we allow a maximum of twenty.

Table 8.6 summarizes the main differences between the approaches. The column *Domain Op* is related with the way the specialist agents generate the candidate-solutions, i.e., if they follow or not the *domain action* according to the probabilistic Tables 4.5, 4.6 and 4.7. The column *Actions* indicates if the manager agents follow or not the action suggested by the learning algorithm when choosing from the candidate-solutions and column *Filter* indicates if the manager agents filter the candidate-solutions according to the domain operator (Tables 4.4, 4.6) suggested by the learning algorithm. The other columns are self-explanatory.

Table 8.6 Approaches comparison

Approach	Learning	CS	Domain Op	Strategy	Actions	Filter
TSA	No	—	No	—	No	No
FB10	No	10	Yes	Best	No	No
Q10-Min	QL	10	No	Min	Yes	No
Q10-Best	QL	10	No	Best	Yes	No
Q10-Best-Filter	QL	10	No	Best	Yes	Yes
Q20-Best	QL	20	No	Best	Yes	No
S10-Best	SR	10	No	Best	Yes	No
S10-Best-Filter	SR	10	No	Best	Yes	Yes
S20-Best	SR	20	No	Best	Yes	No
Q10-Best-V2	Improved	10	No	Best	Yes	No
Q20-Best-V2	Improved	20	No	Best	Yes	No

Finally, the equations and parameters used in the learning mechanism are presented in Table 8.7. The reason to choose these parameter values instead of any others, comes from the experience we had applying these learning algorithms in other projects in LIAAC/NIADR[2].

[2] Artificial Intelligence and Computer Science Laboratory the Distributed Artificial Intelligence and Robotics Group

Table 8.7 Equations and parameters used in the learning mechanism

Parameters/Equations	Values
(T) Temperature:	12
(α) Learning rate:	0,9 (β=0,05)
(γ) Discount factor:	0.6
Q-Value update formula for Q-Learning:	Equation 6.31
Boltzmann Exploration formula used Q-Learning:	Equation 6.30
Q-Value update formula for SARSA:	Equation 7.23

8.2 Results and Discussion

The results of the experiments for the approaches defined in the previous section are summarized in Tables 8.8, 8.9 and 8.10 presented at the end of this section. In Chapter 9 we will use some of these results to draw conclusions regarding the hypotheses we have elicited in Section 1.3.2. Here, we are going to discuss these results trying to answer the following questions:

1. Manual versus automated processes: Can we say that automated processes are better?
2. Are the approaches based on *GQN* (our proposed protocol) better than the *TSA* (traditional sequential approach)?
3. What is the impact on the final solutions of having more candidate-solutions to choose from?
4. What is the impact of filtering the candidate-solutions considering the relation between the event that caused the problem and the domain operator used to solve it, as indicated by the human AOCC operators (see Section 4.6)?
5. Which of the learning mechanisms used, i.e., Q-Learning and SARSA, has the best behavior when applied to this application domain?
6. Which of the manager strategies to select from the candidate-solutions is better, *Nearest Strategy* (Section 7.6.3.4) or *Best Utility Strategy* (Section 7.6.3.5)?
7. Are the *GQN*-based approaches able to select the best integrated solutions when compared with the current typical approach of the AOCCs?

Regarding the ***first question***, any of the automated approaches is significantly better than the manual process of *TAP-AOC*. This was expected. It is reasonable to conclude that, having computer-based tools capable of generating more candidate-solutions and with access to more information, will lead to better solutions. It is not easy to generalize this result to say that *any* automated approach will be better than *any* manual one. However, as we stated in Section 1.2, studies have estimated that irregular operations can cost between 2% and 3% of the airline annual revenue (Chen *et al.*, 2010) and that a better recovery process could result in cost reductions of at least 20% (Irrang, 1996).

Looking at the chart in Figure 8.1 and considering the aircraft and crew average cost reduction, even with the *TSA* approach (the one that implements the sequential approach of disruption management) it is possible to have a cost reduction of 8,62%. Using the approaches based on the *GQN protocol* this cost reduction varies between 17,82% and 58,45%. These results are in accordance with the results presented in the studies of (Irrang, 1996).

Considering the specific case of TAP Portugal, that according to the 2010 Annual Report (TAP, 2011) had an annual revenue of € 1.986,3M, the irregular operations could have a cost between € 39,7M to € 59,5M. Looking at the chart in Figure 8.2, a better recovery process for TAP, based on any of the approaches, could mean a cost reduction between € 4,28M to € 28,99M.

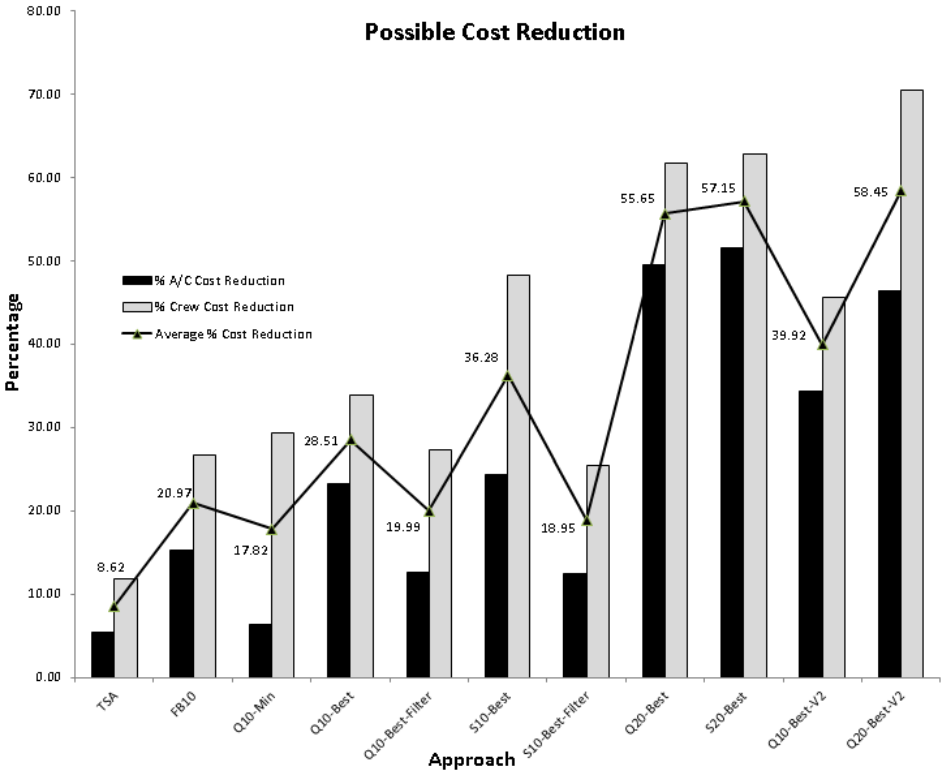


Fig. 8.1 Possible % of Cost Reduction

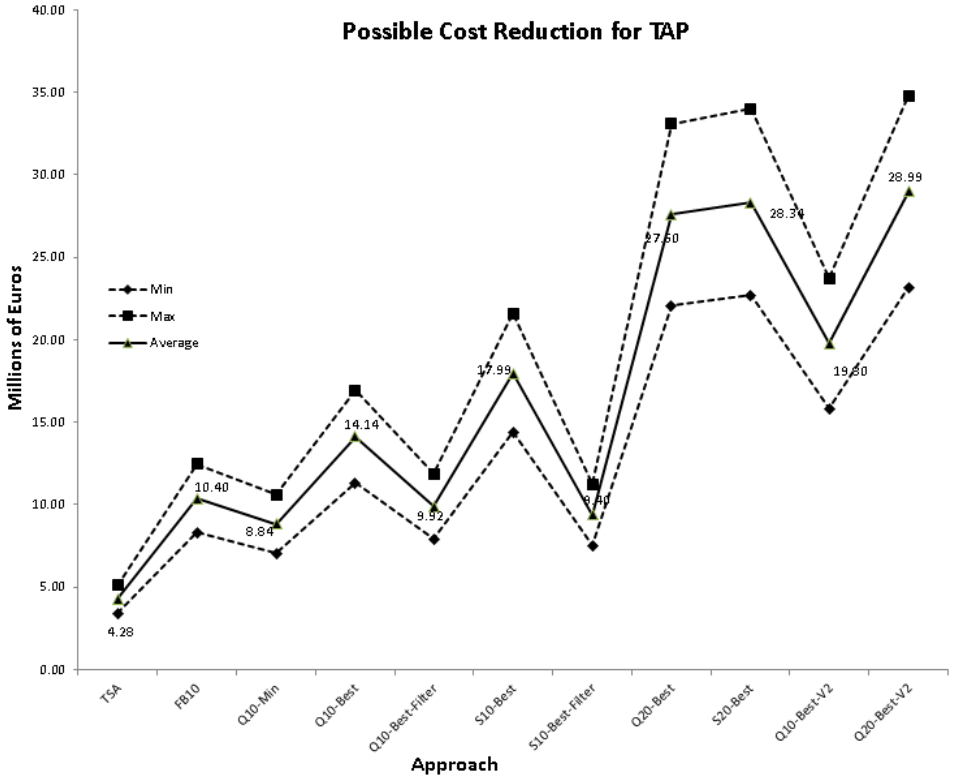


Fig. 8.2 Possible Cost Reduction for TAP considering 2010 revenue

Regarding the ***second question***, in general, the approaches based on *GQN* are better. However, for some metrics, the *TSA* approach achieves better results. For example, looking at the chart of the *Average Flight Departure Delay Recovery Ratio (FDrcv)* (Figure 8.3), it is possible to see that *TSA* achieves better results than *Q10-Min* and *S10-Best-Filter*, being the last four approaches the best ones. However, these results are only possible at the expenses of higher costs, as it is possible to see in the *Average of Combined Cost (FC, CC, PC) per minute* chart in Figure 8.4.

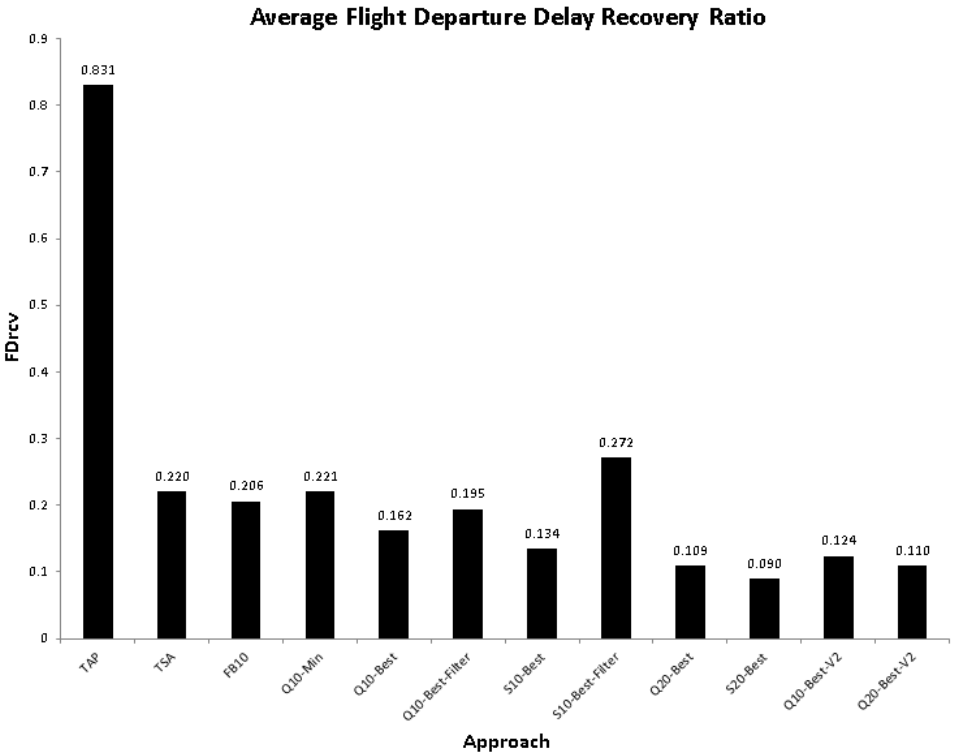


Fig. 8.3 Average Flight Departure Delay Recovery Ratio (FDrcv)

Another metric where *TSA* achieves better results than some of the *GQN*-based approaches, is in the *Average Utilitarian Social Welfare (Usw)*. As it is possible to see in Figure 8.5 only the last four approaches are better than *TSA*. The reason for this behavior can be found in one of the most important charts we can plot from the results, i.e., the *Agent's Utilities and* $\Delta(optimal)$ chart in Figure 8.6. Here, it is possible to see that *TSA* achieves very good utilities for the *aircraft* and *crew* manager agents and a not so good utility for the *passenger* manager. This is a consequence of the sequential approach and, since the *Usw* is the sum of the utilities, the value obtained by the *TSA* is explained. However, this result should also be analyzed together with the $\Delta(optimal)$ one and, as it is possible to see, the *TSA* has the worst result between them, meaning that it produces solutions that are not close to the optimal.

As a conclusion regarding the ***second question*** we can say that the *GQN* approaches are better, specially the last four, i.e., *Q20-Best*, *S20-Best*, *Q10-Best-V2* and *Q20-Best-V2*.

Regarding the ***third question***, we can definitely say that approaches that generate more candidate-solutions are typically better than the other approaches. If we look at the results of *Q20-Best*, *S20-Best* and *Q20-Best-V2* (Tables 8.8, 8.9 and 8.10) we can see that, in all metrics except the *NR*

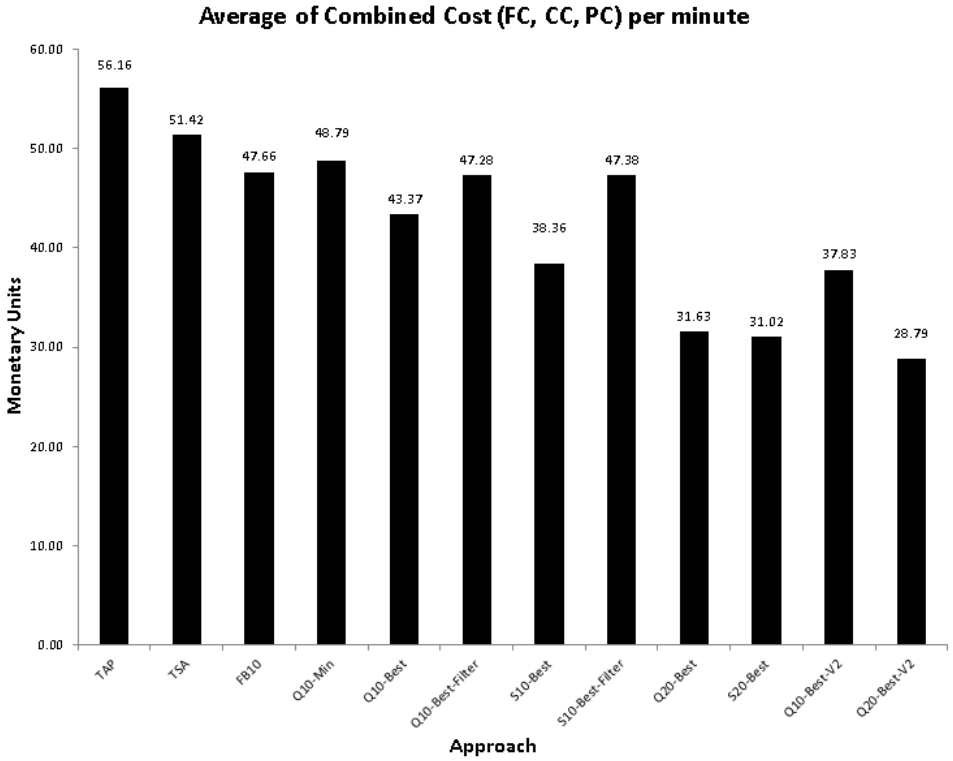


Fig. 8.4 Average of Combined Cost (FC, CC, PC) per minute

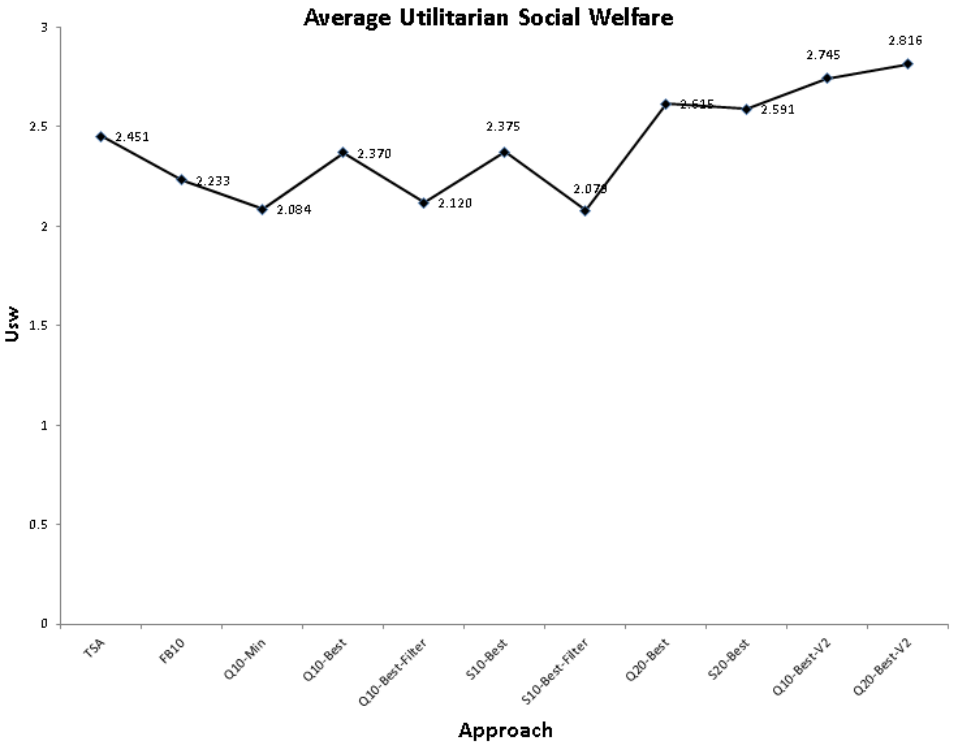


Fig. 8.5 Average Utilitarian Social Welfare (Usw)

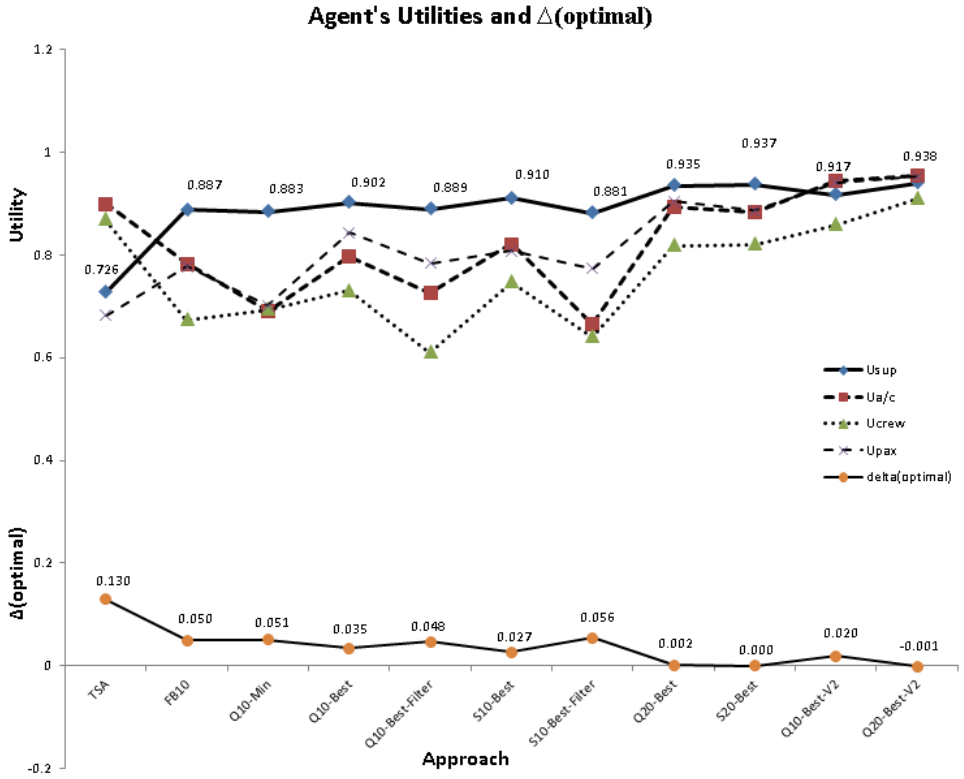


Fig. 8.6 Agent's Utilities and $\Delta(optimal)$

(number of rounds to reach an agreement) and *Mprb* (number of messages per problem), these approaches perform better. As expected this shows that if there are more candidate-solutions to choose from, then the possibility of getting better solutions is higher.

Regarding the ***fourth question***, we can say that what seemed to be a good idea to follow, i.e., to use the adequate *domain operator* to solve a problem according to the type of event that cause the problem (for example, in the case of a problem caused by a maintenance event, 80% of the times the best thing to do is to exchange the aircraft with another), does not work very well when using an automated approach. As we stated previously, this idea resulted from the interviews performed to the human AOCC operator (see Section 4.6) regarding the processes and *rules-of-thumb* they use to solve the problems. The approaches *FB10*, *Q10-Best-Filter* and *S10-Best-Filter* follow this idea.

As it is possible to see in Figure 8.3, regarding delays, the *Q10-Best-Filter* approach only gets better results than the *TSA*, *FB10* and *Q10-Min* approaches (the *S10-Best-Filter* is even worst). The same is true regarding the costs as it is possible to see in Figure 8.4. Even regarding the $\Delta(optimal)$ solution (Figure 8.6), these three approaches (*FB10*, *Q10-Best-Filter* and *S10-Best-Filter*) are only clearly better than the *TSA* one. Finally, regarding the *Utilitarian Social Welfare* (Figure 8.5) these three approaches are amongst the worst ones.

These results are partially corroborated by the results of the *ITGqual* metric (see Tables 8.8, 8.9 and 8.10), where the approaches that are closer to the *preferences* of the AOCC human operators, i.e., *FB10* and *S10-Best-Filter*, are amongst the ones referred previously. The only positive result is regarding the *Number of Rounds to Reach an Agreement (NR)* metric, where only *Q10-Min* has a better result.

This is explained by the fact that filtering the candidate-solutions by the *domain operator* reduces the number of choices and, consequently, the chance of getting better solutions.

Regarding the ***fifth question*** and comparing the approaches where the only difference is the learning mechanism used, i.e., approaches *Q10-Best*, *S10-Best*, *Q20-Best* and *S20-Best*, we can see that, regarding the metrics analyzed so far (see charts Figure 8.3, 8.4, 8.6 and 8.5) the *SARSA* learning mechanism tends to be better, specially in the cases where the approaches generate less candidate-solutions. When the approaches generate more candidate-solutions, *SARSA* is still better but by a shorter difference. Only in the *NR* metric the *Q10-Best* obtained a better result.

The explanation for this slightly better behavior seems to rely on the way that each learning mechanism updates the *Q(s,a) values*. In Q-Learning, the value used in the update corresponds to the maximum value that it is possible to get from the arrived state, i.e., follows an *off-policy* update. On the contrary, SARSA updates the *Q(s,a) values* following an *on-policy* update. There is also a difference between the *Softmax* strategy used in Q-Learning to select the actions and the $\varepsilon - greedy$ strategy used in SARSA. It seems that the *on-policy* update characteristic of SARSA plus the use of $\varepsilon - greedy$ policy in our SARSA implementation results better than our implementation of Q-Learning, when we have a set of candidate-solutions to choose from that are not static, i.e., which changes from round to round. Nevertheless, we can not state that one is better than the other, because there are many parameters in both mechanisms that we did not experiment with.

Regarding the ***sixth question***, we can state that the *Best Utility* strategy (see Section 7.6.3.5) is the best one to use. The strategy based on the *Selection of the Nearest Solution* (see Section 7.6.3.4) only gets better results in the *NR* metric, where it is the best one. On the contrary, in the *Usw* metric it is the worst one. Looking at the *Agent's Utilities and* $\Delta(optimal)$ chart in Figure 8.6 we can see that the approach that uses this strategy is only better than *TSA*. This strategy tries to minimize the difference from a new proposal to present in a future round regarding the one presented in the previous one. This seems to be inadequate for an environment that has more cooperative than competitive characteristics, as the one in the disruption management in the AOCC application domain. Nevertheless and since *GQN* is a protocol that can be applied in several types of environments, including more competitive ones, this strategy might be more suitable in such cases.

Finally, regarding the ***seventh question*** there are some evidences that show that the *GQN*-based approaches are able to select the best integrated solutions when compared with the current typical approach of the AOCCs (see Figures 8.6 and 8.5):

1. The utility of the winning proposals (the ones obtained by the supervisor agent) increased a minimum of 21,35% to a maximum of 29,20%, and an average value of 25,28% when compared with the *TSA* approach.
2. Although the individual utilities for aircraft and crew in the *TSA* approach achieve very good results, being very close to each other, the one obtained by the passenger agent is far from those two. Because of that, the *TSA* approach has the worst *Global Fairness (GF)* (variance) of all approaches. All the *GQN* based approaches achieve a better global utility and, at the same time, a more balanced utility for each of the agents that represent each dimension of the problem.
3. Although the *TSA* approach has a *Utilitarian Social Welfare (Usw)* better than some of the *GQN* approaches (for the reasons we already stated), when compared with the four best *GQN* approaches, the *Usw* value for these four *GQN* approaches, increases from a minimum of 5,7% to a maximum of 14,9%, an average of 10,31%, meaning that these approaches lead to more cooperative solutions.

8.3 Chapter Summary

In this chapter we have presented the experiments we have designed and performed as part of the research methodology we follow. We introduced the scenario we have setup, including the characterization of the data used and the problems to be solved. We have formally defined all the metrics used to evaluate the experiments, including metrics related to *Air Transport*, *Negotiation Outcome*, *Solution Quality* and *Protocol Performance*. We have also explained the different approaches or methods, used to solve the problems that are part of the experiments. These approaches include a *manual* one and eleven *automatic* ones. One of the approaches, *TSA - Traditional Sequential Approach* simulated the current process of disruption management followed by most of the research work proposed in this area by other researchers and, also, by the current AOCCs (see Section 4.6). Ten of the automated approaches are based on our proposed GQN protocol and MAS (see Chapters 6 and 7).

The results were presented and a discussion about their interpretation was then followed. The main conclusions are:

- The automated approaches are better than the manual processes used in *TAP-AOC*.
- Although the *TSA* approach achieves good results for some of the metrics, the *GQN* based approaches are generally better.
- A *common sense* evidence is demonstrated by experimentation: provided that the selection mechanism is rational, i.e., that always maximizes its utility, having more candidate-solutions to choose from generally leads to better solutions.
- The idea of using the adequate *domain operator* to solve a problem (see the fourth question in Section 8.2), that resulted from interviewing the human users of the AOCC regarding their current practice in solving the problems, proved inadequate for use in automated processes.
- When comparing the two learning mechanisms used, i.e., Q-Learning and SARSA, the latter was slightly better in some conditions. However, we can not conclude definitively that one is better than the other.
- The *GQN*-based approaches typically achieve better integrated solutions.
- Four of the *GQN*-based approaches, namely, *Q20-Best*, *S20-Best*, *Q10-Best-V2* and *Q20-Best-V2* achieve the best results.

In the next Chapter we conclude the presentation of this thesis' work by confronting the hypotheses we have presented in Chapter 1 and giving a summary of the main contributions of our work. The limitations of our proposal as well as the future work are also presented.

Table 8.8 Results for approaches *TAP-AOC*, *TSA*, *FB10* and *Q10-Min*

Metric	TAP-AOC	TSA	FB10	Q10-Min
Air Transport Related				
$\overline{FD}$ (min)	36,05	6,15	5,55	6,16
$\overline{CwD}$ (min)	—	4,75	13,91	13,44
$\overline{PD}$ (min)	41,00	28,25	8,06	7, 29
$\overline{FC}$ (m.u.)	2081	1967	1618	1786
$\overline{CC}$ (m.u.)	2158	1904	1472	1469
$\overline{PC}$ (m.u.)	—	2825	2022	1979
$\overline{FDrcv}$ (ratio)	0,831	0,220	0,206	0,221
$\overline{FCrcv}$ (ratio)	1,087	1,028	0,921	1,019
$\overline{CCrcv}$ (ratio)	1,070	0,944	0,785	0,756
$\overline{FCmin}$ (m.u. per min)	47,949	45,315	45,265	49,951
$\overline{CCmin}$ (m.u. per min)	49,274	43,862	41,173	41,078
$\overline{PCmin}$ (m.u. per min)	—	65,092	56,555	55,349
$\overline{FD15min}$ (#)	18	2,5	2,8	3,0
$p\left(\overline{FD15min}\right)$ (%)	36,73	5,1	5,7	6,1
Negotiation Outcome Related				
$\overline{Usup}$ [0,1] >	—	0,726	0,887	0,883
$\overline{Ua/c}$ [0,1] >	—	0,9	0,783	0,691
$\overline{Ucrew}$ [0,1] >	—	0,869	0,673	0,693
$\overline{Upax}$ [0,1] >	—	0,682	0,777	0,701
$\overline{Usw}$ [0,3] >	—	2,451	2,233	2,084
$\overline{\Delta\left(optimal\right)}$ [0,1] <	—	0,130	0,050	0,051
GF (var) <	—	0,009	0,003	0,000
Solution Quality Related				
$A/Cqual$ [0,1]	—	—	0,288	0,175
$CRWqual$ [0,1]	—	—	0,100	0,091
$PAXqual$ [0,1]	—	—	0,020	0,240
$ITGqual$ [0,1]	—	—	0,136	0,169
Protocol Performance Related				
$\overline{NR}$ (#)	—	—	5,12	1,95
$\overline{NT}$ (sec)	—	—	0,09	1,21
$\overline{ST}$ (sec)	—	51,00	0,07	0,09
$\overline{Msg}$ (#)	—	760	30538	23482
$\overline{Mprb}$ (msg per prob)	—	38,00	623,22	479,23

Table 8.9 Results for approaches *Q10-Best*, *Q10-Best-Filter*, *Q20-Best* and *S10-Best*

Metric	Q10-Best	Q10-Best-Filter	Q20-Best	S10-Best
Air Transport Related				
$\overline{FD}$ (min)	4,55	5,28	2,91	3,95
$\overline{CwD}$ (min)	11,69	14,83	8,48	10,69
$\overline{PD}$ (min)	6,13	6,94	3,40	6,48
$\overline{FC}$ (m.u.)	1449	1715	995	1426
$\overline{CC}$ (m.u.)	1327	1422	804	1040
$\overline{PC}$ (m.u.)	1876	1935	1592	1648
$\overline{FDrcv}$ (ratio)	0,162	0,195	0,109	0,134
$\overline{FCrcv}$ (ratio)	0,835	0,949	0,549	0,823
$\overline{CCrcv}$ (ratio)	0,708	0,778	0,409	0,553
$\overline{FCmin}$ (m.u. per min)	40,518	47,961	27,839	39,888
$\overline{CCmin}$ (m.u. per min)	37,125	39,764	22,499	29,096
$\overline{PCmin}$ (m.u. per min)	52,477	54,120	44,538	46,094
$\overline{FD15min}$ (#)	2,0	2,0	0,0	1,0
$p\left(\overline{FD15min}\right)$ (%)	4,1	4,1	0,0	2,0
Negotiation Outcome Related				
$\overline{Usup}$ [0,1] >	0,902	0,889	0,935	0,910
$\overline{Ua/c}$ [0,1] >	0,797	0,726	0,894	0,821
$\overline{Ucrew}$ [0,1] >	0,730	0,610	0,817	0,746
$\overline{Upax}$ [0,1] >	0,844	0,784	0,904	0,808
$\overline{Usw}$ [0,3] >	2,370	2,120	2,615	2,375
$\overline{\Delta\left(optimal\right)}$ [0,1] <	0,035	0,048	0,002	0,027
GF (var) <	0,002	0,005	0,002	0,001
Solution Quality Related				
$A/Cqual$ [0,1]	0,145	0,180	0,169	0,220
$CRWqual$ [0,1]	0,098	0,110	0,093	0,095
$PAXqual$ [0,1]	0,263	0,231	0,244	0,185
$ITGqual$ [0,1]	0,169	0,174	0,169	0,167
Protocol Performance Related				
$\overline{NR}$ (#)	3,78	2,44	4,71	5,24
$\overline{NT}$ (sec)	0,97	4,73	1,07	2,88
$\overline{ST}$ (sec)	0,08	0,50	0,06	0,20
$\overline{Msg}$ (#)	23761	23380	23692	23844
$\overline{Mprb}$ (msg per prob)	484,93	477,15	483,51	486,61

Table 8.10 Results for approaches *S10-Best-Filter*, *S20-Best*, *Q10-Best-V2* and *Q20-Best-V2*

Metric	S10-Best-Filter	S20-Best	Q10-Best-V2	Q20-Best-V2
Air Transport Related				
$\overline{FD}$ (min)	6,79	2,91	3,56	2,95
$\overline{CwD}$ (min)	13,61	8,48	10,79	7,75
$\overline{PD}$ (min)	8,48	3,40	4,76	3,52
$\overline{FC}$ (m.u.)	1669	995	1273	1006
$\overline{CC}$ (m.u.)	1503	804	1075	619
$\overline{PC}$ (m.u.)	1911	1592	1709	1463
$\overline{FDrcv}$ (ratio)	0,272	0,109	0,124	0,110
$\overline{FCrcv}$ (ratio)	0,952	0,549	0,714	0,583
$\overline{CCrcv}$ (ratio)	0,798	0,409	0,583	0,315
$\overline{FCmin}$ (m.u. per min)	46,674	27,839	35,594	28,149
$\overline{CCmin}$ (m.u. per min)	42,024	22,499	30,076	17,310
$\overline{PCmin}$ (m.u. per min)	53,442	44,538	47,807	40,913
$\overline{FD15min}$ (#)	7,0	0,0	0,3	0,0
$p\left(\overline{FD15min}\right)$ (%)	14,3	0,0	0,7	0,0
Negotiation Outcome Related				
$\overline{Usup}$ [0,1] >	0,881	0,935	0,917	0,938
$\overline{Ua/c}$ [0,1] >	0,666	0,894	0,945	0,955
$\overline{Ucrew}$ [0,1] >	0,640	0,817	0,859	0,908
$\overline{Upax}$ [0,1] >	0,774	0,904	0,941	0,953
$\overline{Usw}$ [0,3] >	2,079	2,615	2,745	2,816
$\overline{\Delta\left(optimal\right)}$ [0,1] <	0,056	0,002	0,020	-0,001
GF (var) <	0,003	0,002	0,002	0,000
Solution Quality Related				
$A/Cqual$ [0,1]	0,158	0,169	0,196	0,213
$CRWqual$ [0,1]	0,066	0,093	0,088	0,086
$PAXqual$ [0,1]	0,185	0,244	0,264	0,235
$ITGqual$ [0,1]	0,137	0,169	0,183	0,178
Protocol Performance Related				
$\overline{NR}$ (#)	2,13	4,71	4,98	5,37
$\overline{NT}$ (sec)	1,28	1,07	0,64	0,64
$\overline{ST}$ (sec)	0,09	0,06	0,05	0,05
$\overline{Msg}$ (#)	23309	23692	23532	23569
$\overline{Mprb}$ (msg per prob)	475,69	483,51	480,25	481,00

Part III
Conclusions

Part III - Conclusions presents the conclusion of our work. This part has only one chapter.

In **Chapter 9 - Conclusions**, we provide the results regarding the hypotheses presented in Section 1.3.2 and a critical appreciation regarding the main contributions presented in Section 1.4. This chapter also identifies the limitations of our proposal and the future work. The chapter ends with a brief remark regarding the application of our work in real world companies.

Chapter 9
Conclusions

Abstract In Chapter 8 we presented the experiments designed and performed to test our hypotheses, including the results obtained for each of the tested approaches. In this chapter we conclude the work presented in this thesis. Besides restating the problem we address and the motivations behind it, the most important part is related to the conclusions we have reached regarding the hypotheses and the expected results. The main contributions of our work are presented and we conclude by listing the limitations of our work and the future lines of research. A final remark regarding the application of our work in real world companies is also included.

9.1 Overview

In this thesis we address the problem of *Disruption Management* in Airline Operations Control, i.e., the management of unexpected events that might affect flights, causing departure or arrival delays, minimizing delays in an effort to contain the costs they originate. We do that following a *problem-oriented* line of research having, at the same time, an *out-of-the-box* thinking.

To tackle this problem we used a *Distributed*, *Decentralized*, *Integrated* and *Dynamic* approach. The approach is *Distributed* because it allows a functional, spatial and physical distribution of the roles and of the environment (i.e., the resources available); it is *Decentralized* because some decisions are made at different nodes of the agents' network; it is *Integrated* because it simultaneously considers the main dimensions of the problem: aircraft, crew and passengers; and it is *Dynamic* because, in real time, several agents are making changes in the environment reacting to the constant change in the real scenario.

In order to deal with this class of problems, we used the Multi-Agent System (MAS) paradigm that, according to many authors (Luck *et al.*, 2005; Ossowski, 2013; Wooldridge, 2009a; Elamy, 2005), allows to model systems that include software agents, representing roles or functions as well as the surrounding environment, including interactions amongst the agents. This view makes this paradigm not only important as a research paradigm but, also, a practical approach to solve real world problems.

Studies have estimated that airline irregular operations can cost between 2% and 3% of the airline annual revenue (Chen *et al.*, 2010) and that a better recovery process could result in cost reductions of at least 20% of its irregular operations (Irrang, 1996). If we consider the specific case of TAP Portugal, that according to the 2010 Annual Report (TAP, 2011) had an annual revenue of €1.986,3M, its irregular operations may have had an estimated cost between €39,7M to €59,5M. A better recovery process could thus mean a cost reduction of its irregular operations between €7,94M to €11,9M. This was, for us, one of the main motivations behind the research work presented in this thesis.

The rest of this chapter is structured as follows. In Section 9.2 we conclude about the validity of the hypotheses and expected results we have listed in Section 1.3. In Section 9.3 we provide a critical analysis regarding the main contributions of our work as presented in Section 1.4, stating

clearly what was and was not accomplished during this work. The limitations of our proposal and the future work are presented in Section 9.4 and we conclude with a final remark in Section 9.5.

9.2 What About the Hypotheses?

Following our research methodology as described in Section 1.3 it is time to perform step 6 of the methodology, i.e., after collecting the data from the experiments and after performing an interpretative analysis of the results (see Chapter 8) we are able to perform a critical analysis of our work, reaching conclusions about the hypotheses and expected results as presented in Section 1.3.2 and 1.3.3, respectively. The conclusions are:

Regarding the first hypothesis we expected to obtain a *MAS Architecture* that allows to:

1. *Distribute roles* to agents according to the specific dimensions of the problem (aircraft, crew and passenger) and required expertise (*functional distribution*), contemplating at the same time, the individual goals and preferences leading to decentralization (*autonomy and privacy*).
2. *Distribute data* to different databases (*spatial distribution*).
3. *Distribute the agents and data* to different computers and locations (*physical distribution*).
4. *Increase scalability* by adding, removing and changing roles, agents and environment according to the AOCC's organization size, needs and policy.
5. *Represent resources or agents* that do not belong to the airline company.

In Chapter 7 we have conceptualized a MAS architecture that complies with all of these requirements and we have developed a system called MASDIMA according to those requirements (see Appendix A). We designed and performed experiments that allowed to evaluate the functional and spatial distribution as well as the scalability referred in items 1, 2 and 4 above. We did not evaluate the physical distribution and the representation of resources or agents external to the airline, as referred in items 3 and 5 above. Regarding the former, the JADE specifications (Fabio Bellifemine & Greenwood, 2007) allow that kind of distribution and since we have developed our system with this framework, it should be straightforward to physically distribute MASDIMA among different computers. The only impact we expect regarding the experiments we have done is an increase in the *Negotiation Time (NT)* and in the *Search Time (ST)*, due to the additional communication overhead. Regarding the representation of external resources or agents, we have conceptualized a set of advanced features in the MASDIMA architecture (Sections 7.4 and 7.8) that shows this kind of representation.

Considering the reasons above, *we accept the first hypothesis*.

Regarding the second hypothesis we expected to obtain a *negotiation protocol* that would:

1. *Increase the Social Welfare* (the sum of agent's individual utilities (Sandholm, 1999)) of the manager agents, allowing also the organizer agent to increase its utility. This will contribute to a better integrated solution.
2. *Balance the utility* of each manager agent, contributing to a more equally distributed importance of each dimension of the problem without implying a decrease on the organizer agent utility.
3. *Allow the autonomy and privacy* of the agents regarding their goals and preferences.

4. *Reach an agreement in fewer rounds* (and, thus, reducing the time to get a solution) reacting, at the same time, to changes in the environment.
5. *Achieve all the above without compromising the goals* of the organizer and manager agents.

In Chapter 6 we have conceptualized the *GQN* protocol that complies with all the items above. We have applied this protocol to disruption management in airline operations as explained in Chapter 7 and implemented it in the MASDIMA system (Appendix A). We designed and performed experiments that validate all of the above 5 items. Regarding item 1 the results (Tables 8.8, 8.9 and 8.10) show that the *Utilitarian Social Welfare (Usw)* increases as well as the *Supervisor Utility (Usup)*. This is specially true for the last four approaches tested. Regarding item 2 the results show that the *Global Fairness (GF)* balances the utility of the managers without decreasing the supervisor utility. Actually, in some cases it even increases that utility. Item 3 is contemplated in the protocol conceptualization and implementation. Regarding item 4 the results for the metric *Negotiation Rounds (NR)* show that the protocol is able to reach an agreement in approximately two to five rounds, depending on the approach used. Looking to the agent's utilities it is possible to see that the approaches that use learning, tend to react better to changes in the environment and, as such, converge better to the supervisor agent preferences. Finally, regarding item 5 the results for the utilities of the agents (*Usup*, *Ua/c*, *Ucrew* and *Upax*) as well as the results of the delays (*FDrcv*) and costs (*FCmin*, *CCmin* and *PCmin*), show that all of the above are accomplished without compromising the goal of minimizing the delays and costs.

Considering the reasons above, *we accept the second hypothesis*.

Regarding the third hypothesis we expected to have *problem solving algorithms* for each manager agent that would:

1. *Increase the number of candidate-solutions* for each manager agent to choose from.
2. *Increase the quality of the candidate-solution* for each manager agent, i.e., candidate-solutions that could increase the utility for each manager.

In Section 7.7 we defined the specialist agents that implemented the problem solving algorithms for each of the dimensions of the problem, i.e., aircraft, crew and passenger. These specialist agents were implemented using the Simulated Annealing algorithm (Kirkpatrick *et al.*, 1983) for the aircraft and crew dimension and the Dijkstra Shortest-path algorithm (Dijkstra, 1959) for the passenger dimension. From the experimental results we show that we were able to increase the number as well as the quality of candidate-solutions for each manager to choose from, and that the utility of each manager is thus increased. That is the case of approaches *Q20-Best*, *S20-Best*, *Q10-Best-V2* and *Q20-Best-V2*. Additionally, the implementation of these problem solving algorithms eliminate the repetitive tasks performed by the different roles and, at the same time, avoids the use of *rules-of-thumb* as the major problem solving process.

Considering the reasons above *we accept the third hypothesis*.

Regarding the fourth hypothesis we expected to *integrate new systems into the MAS* that would:

1. *Bring additional information* that allows to improve the quality of candidate-solutions by increasing the utility for each manager agent (this expected result is closely related to the one of the *third hypothesis* above).
2. *Bring other means of transportation* allowing to have more diverse and alternate candidate-solutions and, as such, increase the number of candidate-solutions to choose from.

3. *Provide better and richer* candidate-solutions that contribute for the negotiation mechanism to reach better solutions, i.e., solutions that increase the utility of the organizer agent and the social welfare of the manager agents.

In Chapter 7, specially in Sections 7.3, 7.4 and 7.8 we provided some conceptualization and insights on how other systems could be integrated in the MAS we propose. We did not implement it nor did we design or perform experiments with such functionalities. Nevertheless, some research work regarding these features is being done at the LIACC/NIADR group. We believe that the implementation of these features as well as the experiments that will later be performed, will prove items 1 to 3 above.

At this moment in time and considering the above we *cannot reach a conclusion* regarding the acceptance or rejection of the *fourth hypothesis*.

9.3 Main Contributions

In Section 1.4 we provide a detailed description of the main contributions achieved with our work. Here we will just recapitulate those contributions, refraining from repeating the details provided before.

The work we have performed to test the hypotheses led to contributions of two types: the ones that are directly linked to those hypotheses, and others that, although not being directly linked, helped to achieve them and comprise significant supplementary work undertaken in this thesis. Both are equally important.

Regarding the former, these are the main contributions:

- A *Multi-Agent System Architecture* that represents the AOCC (Chapter 7). Directly linked to the work done to prove hypotheses one, three and four.
- A specification of a *Generic and Adaptive Negotiation Protocol* (Chapter 6). Directly linked to the work done to prove hypothesis two.
- An implementation of an *Advanced Prototype*, called *MASDIMA*, that includes the above scientific contributions (Chapter 7 and Appendix A). Directly linked to all of the hypotheses.

Regarding the latter, we would like to point out the following main contributions:

- A specification of an *Agent-Oriented Methodology*, called *PORTO* (Chapter 5).
- A method to *Calculate the Quality Costs from the Passenger Point of View* (Section 7.5.2).
- A *System Classification* to classify disruption management systems (Section 3.3).

9.4 Limitations and Future Work

The main limitations of our work are related to the scenario we have setup for the experiments. As we have stated in Section 8.1.1, there are two limitations.

The first one is related to the seasonal nature of the air transport activity. Ideally, to be able to fully test the hypotheses we should use one year of real data from the airline operations plan of TAP Portugal. Unfortunately, one full year of data corresponds to several TB (terabytes). Preparing that data to be used by our *test laboratory*, i.e., the MASDIMA system, is a huge undertaking. To minimize the impact of this limitation we decided to use the September 2009 data since it has

characteristics similar to the average of one year of operations. Therefore, despite the limitations inherent to the quantity of data used, we think that the results can be extrapolated with some confidence to a year of operations.

The second one is related to the *dynamics* of the information. In a real operation environment, the information is not *static*, i.e., it might change during the disruption management process. For example, a new event might affect a resource that is being considered as a possible solution to a problem, making the solution infeasible and reducing the number of possible solutions to choose from. To minimize the impact of this limitation and to try to capture this dynamics, we limited the number of candidate-solutions generated by each agent to a maximum and, randomly, made sure that from one negotiation iteration to another there is a small subset of candidate-solutions that is different or unavailable.

Regarding future work, there are several interesting lines of research that, although we have referred them in some of the chapters, are important enough to be pointed out again here[1].

Regarding the *GQN* protocol, we expect to improve the following group of features:

- *Open Environments*: The GQN protocol has been used in closed environments. We want to make the protocol more suitable to be used in open environments, e.g., by including an Ontology Service component, similar to the one of (Malucelli *et al.*, 2006), which would allow using agents made by different designers to make the match between problem dimensions and agent competences in a more dynamic way. This component would also allow to define more dynamically the negotiation object, i.e., the set of attributes to be used in the negotiation.
- *Organizer Agent Learning*: Our protocol already includes adaptable strategies through the inclusion of learning mechanisms during proposal formulation on the respondent agents (RA). We want to include learning in the organizer agents (OA). We believe that the *OA* could "learn with the past" and use this information before starting a new negotiation. Additionally, since we plan to include *Human-in-the-loop* features (see Chapter 7) the *OA* could use the feedback information received from the human to adapt its strategy not only before starting a new negotiation but, also, during the negotiation.
- *Argumentation*: The GQN protocol already includes qualitative feedback. However we believe that the inclusion of arguments in our model will support more negotiation scenarios, making it more suitable to be used in more real world applications. We want to change our model to include *argument interpretation*, *argument generation* and *argument selection*.
- *Normative Environment and Trust*: Finally, we want to integrate our protocol in the ANTE framework (Oliveira, 2012; Lopes Cardoso *et al.*, 2013). In doing so, we may take advantage of a normative environment and computational trust, two characteristics that, in our opinion, will enrich the model.

Regarding the MAS architecture, we expect to include the following features:

- *Maintenance Personal Tablet PCs*: Improves the communication between the *Maintenance Services* team and the maintenance personnel at the airport. Less errors when reporting events to the AOCC and better decision making is expected.
- *Passengers Smartphones*: It will allow the passengers to receive alerts regarding disruptions, including new ETD and itineraries. It will also contribute to better decision making and, at the same time, improves the quality of the service provided to the passengers.

[1] It is important to point out that some of the *future work* items are already being pursued by some master degree students at the LIACC/NIADR and other features were presented as research proposals to PhD students of the Doctoral Program in Informatics Engineering (PRODEI).

- *Crew members Tablet PCs*: Pilots and cabin crew can perform their operational tasks better, faster and with less errors. A better and wider integration of information can be achieved.
- *Aircraft Datalink Connection*: Data exchange between the aircraft and ground information systems will allow a proactive action towards avoiding flight arrival delays. It will also contribute to better decision making.
- *Alternative Transportation Datalink*: As it was proved by the 2010 eruption of the Iceland Eyjafjallajökull volcano that caused flight disruptions that took days to be solved, under certain conditions to have alternate itineraries performed by different means of transportation (e.g., bus, train) might be an acceptable solution. This is specially important to improve the disruption management process for the passenger part of the problem.
- *Aircraft and Crew Electronic Market*: Nowadays, it is common for the airline companies to lease an aircraft and crew in some situations (called ACMI). The idea is to propose an electronic market that will allow to contract services regarding these two important resources.

9.5 Final Remark

As we state in Section 1.1 we follow the *problem-oriented* line of research, meaning that our utmost goal is to provide techniques that can be used in real world problems. It is understandable that, after reading the work we present in this thesis, one can ask about the possibility of applying our proposal with success in real world companies, like TAP Portugal, for example. What are the gains? What are the challenges and difficulties? Is it possible to apply the proposal to other application domains besides the airline operations control?

We have shown in Section 8.2 Figure 8.2 that, in the specific case of TAP and considering the 2010 revenues, using the approach proposed by us with lower results, i.e., *Q10-min*[2], TAP could have a cost reduction of €8,84M. Now, suppose that, after applying our proposal in TAP, for whatever reason, we could only get 15% of that value. Even in this case, it would mean a cost reduction of €1,33M approximately, every year. We think that the potential gains compensate the risks that result from implementing our proposal.

The main challenges and difficulties are related with the available data from two points of view: *availability* and *accuracy*. In a system like the one we propose, it is of paramount importance to have access to different types of data, such as: flight information, aircraft maintenance and costs, crew availability and costs, passenger reservations, passengers costs and preferences, passenger booking information, ground handling operations, other airlines schedules, current weather and forecasts, etc. Nowadays, this information exists but it is scattered by different systems and some of those systems are not owned by the airline company. Additionally, it is crucial that this information is available in real-time, which is not always the case.

After solving the problem of data availability, it is necessary to guarantee accurateness, – the data to which the system will have access must match real world data, preferable in real-time or almost real-time. There is a test that the author of this thesis is used to perform. From his office at TAP he is able to see the Lisbon airport runway. Whenever he sees an aircraft landing or taking off, he immediately checks the existing flight information system to see if the landing or take off time is already registered. Sometimes it takes several minutes for this information to appear and,

[2] We are not considering here the *TSA* approach since it simulates the current disruption management process used in the AOCC and was built for comparison purposes only.

many times (for reasons that are out of the scope of this final section), it does not correspond to the real landing or take off time. This is just an example. The same level of accuracy is likely to exist for most data.

As we state in Section 1.1 we tried to keep an *out-of-the-box* thinking when conceptualizing and designing the proposals presented in this thesis. We believe that the proposals presented in Chapters 6 and 7, regarding the negotiation protocol and the system architecture, respectively, are generic enough to be applied in different problems in the same application domain and, also, in other application domains. However, the best answer to this question comes from industry. Since the beginning of this year, EMBRAER[3] is interested in applying the distributed, integrated and dynamic approach we have proposed for the AOCC to a project related with the integration of operations information, maintenance, logistics and prognostics and health monitoring, regarding automatic decision support for aircraft maintenance. They have read some of the papers we have published and contacted us to cooperate regarding this subject.

[3] Embraer - Empresa Brasileira de Aeronáutica (http://www.embraer.com.br/)

Part IV
Appendixes

Part IV - Appendixes includes additional information that, although important, makes more sense to be used as a reference. It includes the following appendixes.

In **Appendix A MASDIMA - Multi-Agent System for Disruption Management**, we describe the main features of the advanced prototype we have developed to test our ideas.

In **Appendix B MASDIMA - Code Examples and Diagrams**, we provide some examples of the JAVA code used to implement some features and a full sequence diagram of the GQN protocol we propose in Chapter 6.

In **Appendix C MASDIMA - Costs Information**, we provide the tables used to calculate the costs regarding the three dimensions of the problem: aircraft, crew and passengers.

Finally, in **Appendix D Delay Codes**, we present the IATA delay code tables as well as the IATA and TAP delay code classification, that support the information provided in Chapter 4.

Appendix A
MASDIMA - Multi-Agent System for Disruption Management

The Multi-Agent System for Disruption Management (MASDIMA) is the advanced prototype we have developed to test our proposal. As we mention in the *Research Methodology* we follow (Section 1.3), MASDIMA is our *test laboratory*.

This system was the result of following the *PORTO Methodology* (Chapter 5). The advanced prototype we show in this appendix, implements the ideas of our proposal for a new approach for disruption management (Chapter 7) and uses the Generic Q-Negotiation Protocol as the decision mechanism (Chapter 6).

To implement the MASDIMA we have used Java[1] as the programming language and JADE (Bellifemine *et al.*, 2004) as the framework for developing the agents and as the runtime environment. Additionally, we have used the NASA Worldwind API[2] as our flight visualization tool.

In Section A.1 we describe the features of the system available through the main user interface. In Section A.2 we present the charts and information related to the negotiation process and the winning solution. Finally, in Section A.3 we give an example that shows the messages exchanged by the agents during the negotiation through a protocol sequence diagram.

A.1 MASDIMA User Interface

MASDIMA runs autonomously requiring little intervention from a human operator. Actually, the system is designed in a way that requires only the intervention of a human operator to validate and give feedback regarding the solutions found automatically. It is prepared to receive information from several sources. Section 7.4 provides more information regarding this subject. Figure A.1 shows the MASDIMA User Interface. It has six main parts as follows:

1. *Flight Monitoring*: This window shows the movement of aircraft in real-time sorted by departure date and time. For example, in the example of Figure A.1, the first line shows flight 813, departing from *MXP* (Milan) to *OPO* (Porto) at 12:15 UTC. This flight has already departed as the column *airborne* indicates. Clicking in a flight will show information regarding the aircraft, crew and passengers (see Part 2) and will center the *globe* in that specific flight (see Part 5). When an event causes a flight delay, the row of the affected flight turns red and the delayed flight will appear in the problem status windows (see Part 3). After the problem is solved, the row turns green.
2. *Flight Information*: After clicking in a flight, this window allows access to information related to the three dimensions of the problem: aircraft, crew and passenger. For example, clicking in tab *aircraft* it is possible to see the tail number of the aircraft as well as several information related to costs, e.g., *maintenance average cost per minute*, *fuel average cost per minute*, etc. Clicking on the other tabs, similar information will appear for that specific dimension.

[1] http://www.java.com

[2] http://worldwind.arc.nasa.gov/java/

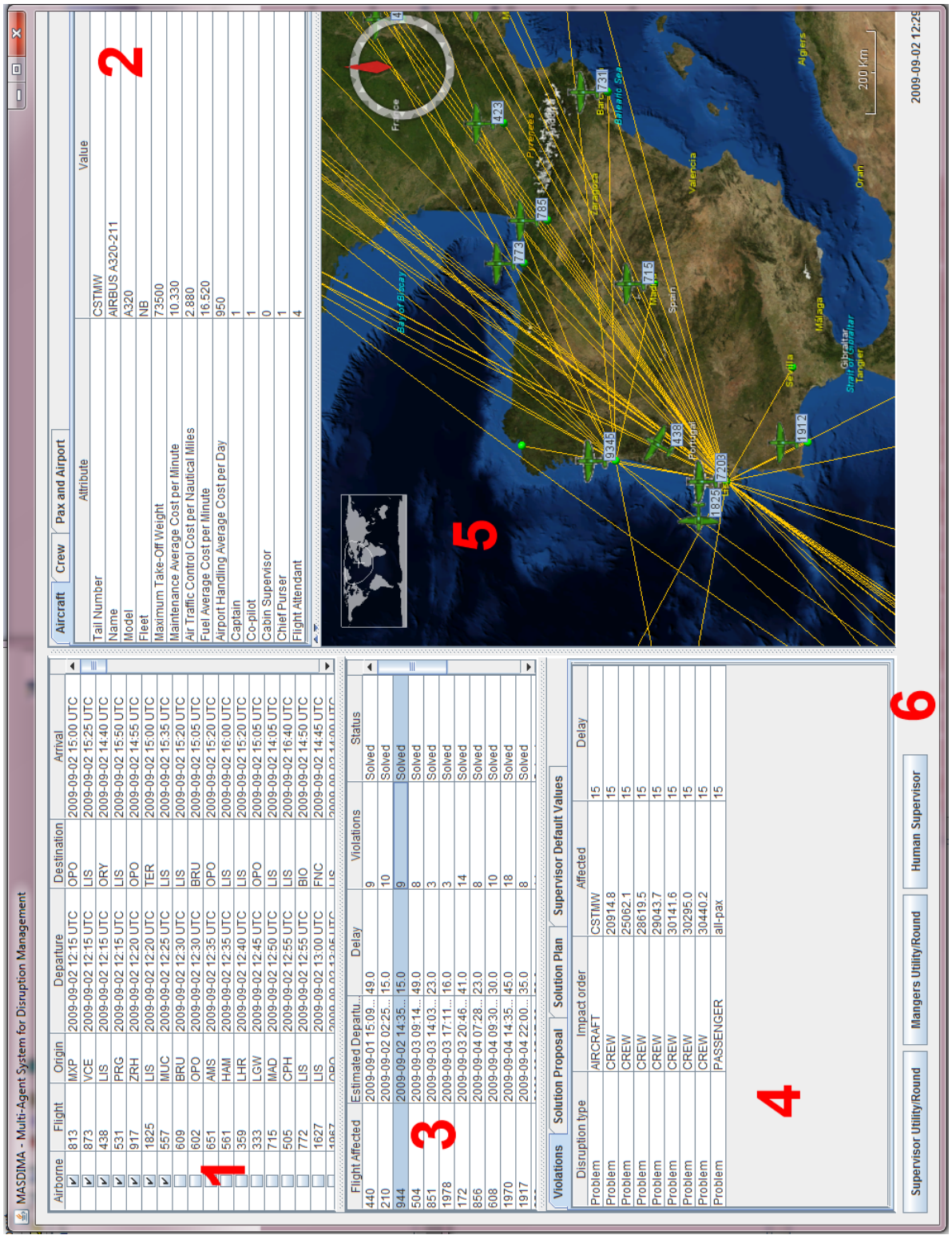


Fig. A.1 MASDIMA Full User Interface

3. *Problem Resolution Status*: Every time an event causes a flight to be delayed, that flight will appear in this window, including the flight number, estimated time of departure, the expected delay (before the problem is solved), the number of violations and the current status of problem (*unsolved* or *solved*). Clicking on a flight, it will be possible to see the specific violations and the default values used by the supervisor agent to solve the problems (even before the problem is solved) and, after the problem is solved, the proposal that won the negotiation and the corresponding plan to be applied (see Part 4).
4. *Solution Information*: In this window it is possible to have access to the specific violations of the flight (e.g. aircraft and crew members affected as well as passengers) through tab *violations* and to the default values used by the supervisor agent during the disruption management process through the tab *Supervisor Default Values*. After the problem is solved, it is possible to have access to the proposed solution through tab *Solution Proposal* (Figure A.4) and the solution plan through tab *Solution Plan* (Figure A.5). More information about the solution found will be given in Section A.2.
5. *Flight Visual Representation*: In this window it is possible to have a visual representation of all flights that are being monitored in a earth globe. This information includes the flight number, origin and destination airport and the *orthodromic route* of the flight. If the MASDIMA is connect to a *datalink server* (a server that receives information in real-time from the aircraft) it is possible to see the flight plan route and the real route being used by the aircraft. This visual representation allows several interactions with the user, for example, it is possible to zoom in, zoom out, rotate, etc. All these features are possible due to the use of NASA Worldwind API.
6. *Charts and Human Feedback*: This part allows access to the supervisor utility by round chart (Figure A.2), managers utility by round chart (Figure A.3) and to the human-in-the-loop user interface. More information about the charts is provided in Section A.2. The human-in-the-loop feature is not completely implemented and more information about it is provided in the advanced features Section 7.8.

A.2 Decision Process Information

After choosing a problem that is solved in the *Solution Information* window (Part 3) it is possible to have access to the buttons that appear in Part 6.

Clicking on the button *Supervisor Utility/Round* in Figure A.1 it is possible to access the chart with the Supervisor utilities for each proposal presented by each manager in every negotiation round. Figure A.2 show the chart for flight 608. In red it appears the supervisor utility for the proposal presented by the passenger manager agent. In blue and green, we have the utilities calculated for the proposals presented by the aircraft and crew manager agents, respectively.

Leaving the mouse for a few seconds in one of the points, it will appear a *tooltip* window showing the proposals presented by each agent as well as the information of which agent won the round and the qualitative feedback given by the supervisor to each attribute of the proposals. For example, in Figure A.2 we have the information for the first round. We can see that the winner proposal was presented by the aircraft manager agent with an utility value for the supervisor of 0,894. Continuing with the example, the proposal presented by the crew manager (one that lost the round) received as qualitative feedback $ok, very\_high, ok, low, ok, very\_high$ for the values of the attributes *AC Delay*, *AC Cost*, *CW Delay*, *CW Cost*, *PX Delay* and *PX Cost*, respectively.

Clicking on the button *Managers Utility/Round* it is possible to access a similar chart but, this time, with the individual utilities that each proposal has for each manager (Figure A.3).

The values for the negotiation winning proposal and the correspondent operational plan, are available by selecting the *Solution Proposal* tab and the *Solution Plan* tab, respectively, in the *Solution Information* window (Part 4).

Figure A.4 shows an example of a winning proposal. In blue we have the values for the attributes of the aircraft dimension, in dark green the values for the attributes of the crew dimension and in light green the ones for the passenger dimension. It is also possible to see that this proposal won with an utility of 0,956.

The correspondent operational plan is presented in Figure A.5, using the same color scheme. In this case, the solution plan is to *exchange* aircraft *CSTOB* with aircraft *CS-TOF*, *accept the delayed crew* and to *keep* the passengers in the same flight.

Finally, clicking on the button *Human Supervisor* we will have access to the *human-in-the-loop* user interface. This feature is not implemented in the current version of the prototype. However, a description of it is given in Section7.8. At the present time, a Master Thesis is being developed around this subject.

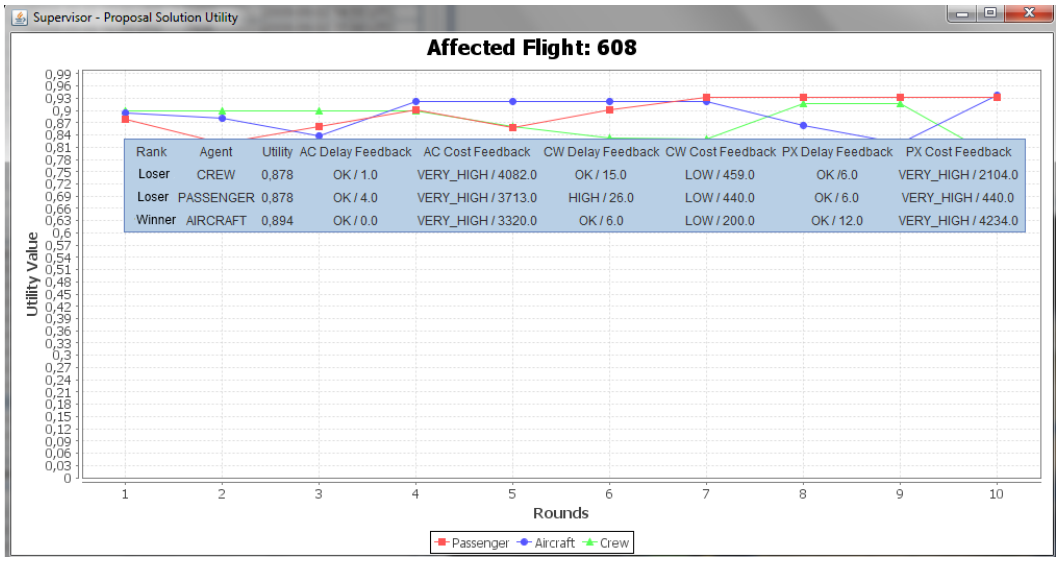


Fig. A.2 Supervisor Proposals Utilities Chart

A.3 Protocol Sequence Chart

As we stated in the beginning of this appendix the MASDIMA advanced prototype implements the GQN protocol we propose in Chapter 6. Additionally, we use the JADE framework not only to develop the agents and protocols but, also, as the runtime environment.

Figure A.6 shows the information we can get regarding the agent's interaction during the negotiation process, using a tool called *Sniffer* from JADE. On the left hand side we can see the implemented agents as well as the ones that are part of the JADE runtime. For example, *df@192.168.1.74:3010/JADE* that is the directory facilitator agent.

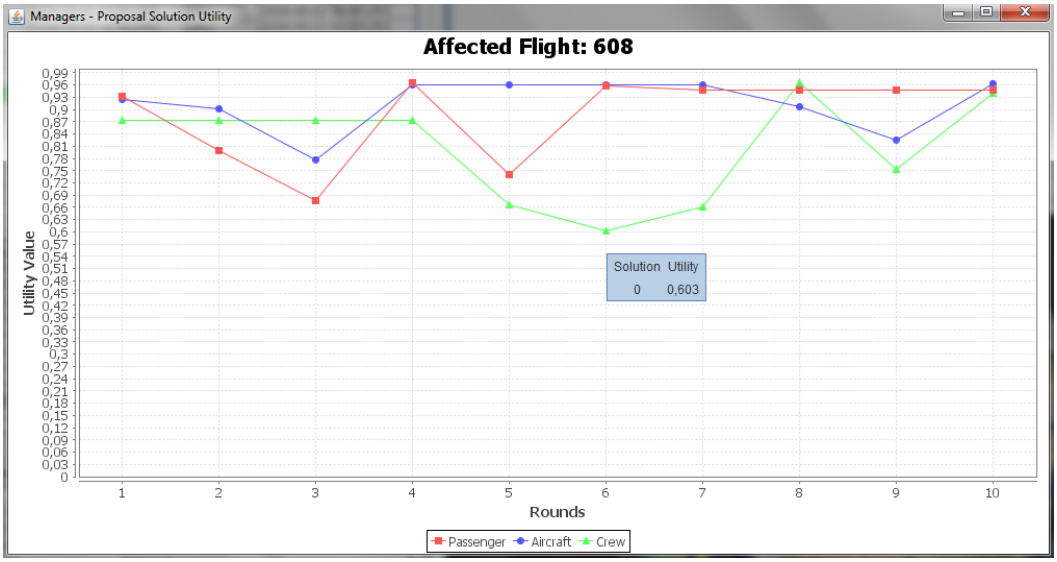
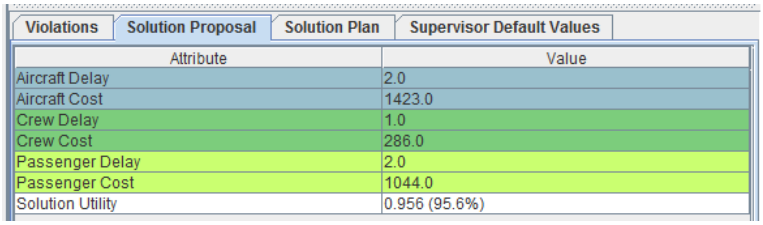
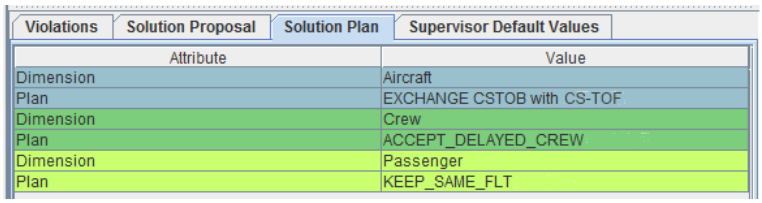


Fig. A.3 Manager Proposals Utilities Chart



Attribute	Value
Aircraft Delay	2.0
Aircraft Cost	1423.0
Crew Delay	1.0
Crew Cost	286.0
Passenger Delay	2.0
Passenger Cost	1044.0
Solution Utility	0.956 (95.6%)

Fig. A.4 Example Solution Proposal



Attribute	Value
Dimension	Aircraft
Plan	EXCHANGE CSTOB with CS-TOF
Dimension	Crew
Plan	ACCEPT_DELAYED_CREW
Dimension	Passenger
Plan	KEEP_SAME_FLT

Fig. A.5 Example Solution Plan

On the right hand side, we can see (partially) the performatives and messages exchanged by the agents, including the interactions with agents that do not participate in the negotiation process, like the *monitoring* and *visualizer* agents (the role of these agents is described in Section 7.4). To make the figure more clear we did not include the interactions with the specialist agents. In the figure they appear as *Other*.

It is also possible to see the negotiation protocols in action. For example, the round square one in Figure A.6 shows the *contract net* protocol between the aircraft and crew manager and their team of specialist agents. Round square two shows the inter-manager negotiation of the *GQN* protocol. Finally, round square three shows the main negotiation of the *GQN* protocol between the managers and the supervisor agent.

More information about these protocols is given in Chapter 6 and in Section 7.6.

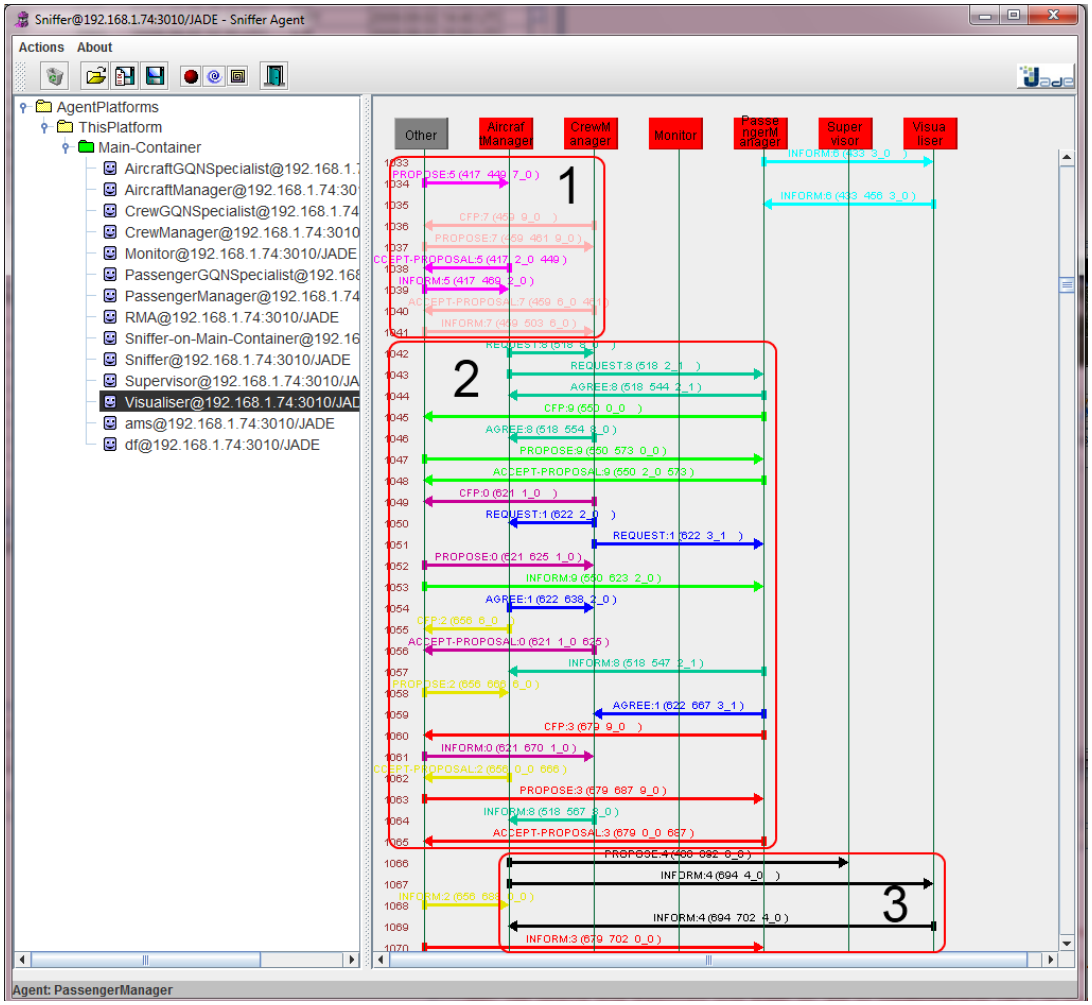


Fig. A.6 MASDIMA GQN Protocol (Partial) Sequence Chart

Acknowledgements We wish to thank to António Pedro Pereira, Leonardo Fraga and Francisca Teixeira for their contribution in implementing parts of the MASDIMA system.

Appendix B
MASDIMA - Code Examples and Diagrams

In this appendix we present some additional diagrams and implementation code examples that have been referred during the thesis. In Section B.1 we present the JAVA code implementation examples and in Section B.2 the diagrams.

B.1 JAVA Code Implementation Examples

The following JAVA code implements the concept *CrewEvent*. As it is possible to see it is a very simple and standard object oriented code.

```
import java.io.Serializable;
public class CrewEvent implements Serializable {
   private String dutyID;
   private String crewNumber;
   private String prngNumber;
   private String openPos;
   private String eventID;

   public String getDutyID() {return dutyID;}
   public String getCrewNumber() {return crewNumber;}
   public String getPrngNumber() {return prngNumber;}
   public String getOpenPos() {return openPos;}
   public String getEventID() {return eventID;}

   public void setDutyID(String d) {dutyID = d;}
   public void setCrewNumber(String d) {crewNumber = d;}
   public void setPrngNumber(String d) {prngNumber = d;}
   public void setOpenPos(String d) {openPos = d;}
   public void setEventID(String d) {eventID = d;}
}
```

The Agent *MonitorAgent*, that implements the roles *RosterCrewMonitor*, *RosterAircraftMonitor* and *PaxMonitor* as well as the *Monitor* and *Update* services associated, are implemented using JADE behaviours *Ticker* and *OneShot* as presented in the following code:

```
import java.core.*;
public class MonitorAgent extends Agent {
   protected void setup() {
      // activate MonitorCrewEvents service
      addBehaviour(new MonitorCrewEventsBehaviour(this, 1000));
      // activate UpdateCrewEvents service
      addBehaviour(new UpdateCrewEventBehaviour());
      // activate MonitorAircEvents service
      addBehaviour(new MonitorAircEventsBehaviour(this, 1000));
      // activate UpdateAircEvents service
      addBehaviour(new UpdateAircEventBehaviour());
```

```
        // activate MonitorPaxEvents service
        addBehaviour(new MonitorPaxEventsBehaviour(this, 1000));
        // activate UpdatePaxEvents service
        addBehaviour(new UpdatePaxEventBehaviour());
}
```

The last partial implementation code example is the implementation of the *MonitorCrewEvents* through the JADE *TickerBehaviour*, as follows:

```
import java.util.Date;
import java.core.*;
public class MonitorCrewEventsBehaviour extends TickerBehaviour {
   public MonitorCrewEventsBehaviour(Agent a, long period) {
      super(a, period);
   }
   // Monitors the crewSignOn and Pairings resources
   protected void onTick() {
      Date dt = new Date("2009-09-10"); // date to monitor
      int crwSlack = 10; //crew slack time
      int pairSlack = 10; //pairing slack time

      // starts the service with the inputs
      CrewEvent ce = monitorResources(dt, crwSlack, pairSlack);
      // if the CrewEvent concept, ie, the output of the service
      // has information it is necessary to start the IP
      // and request a solution
      if (ce != null) {
         // starts the request-solution protocol
      }
   }
   // method that implements the SQL code to get
   // information from the environment
   private CrewEvent monitorResources(Date d, int cs, int ps) {
   CrewEvent ce = new CrewEvent();
   //
   // Perform the SQL code to monitor the resources
   // returning the CrewEvent filled with information
   return ce;
   }
}
```

B.2 Diagrams

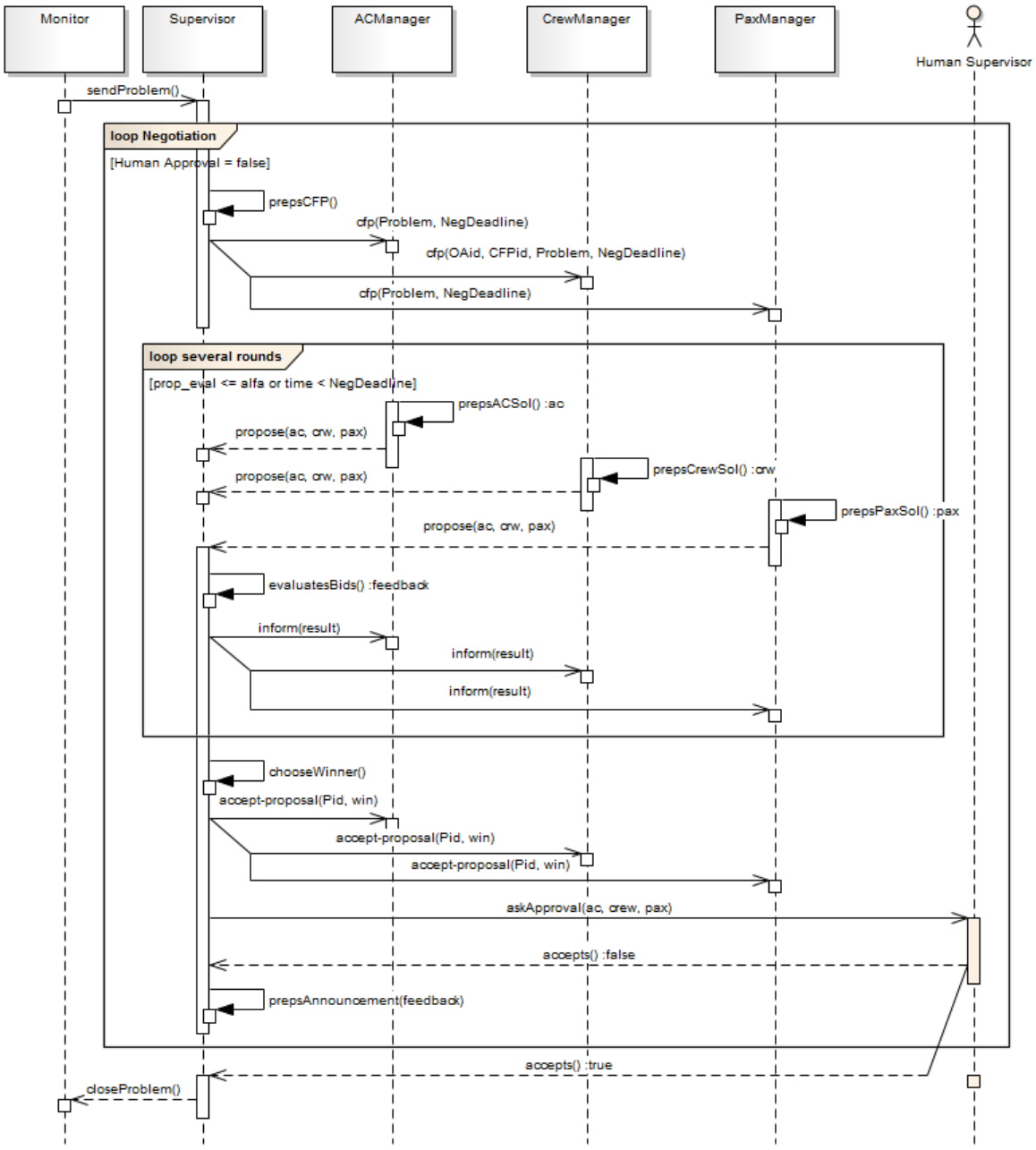


Fig. B.1 GQN Sequence Diagram applied in MASDIMA

Appendix C
MASDIMA - Costs Information

Table C.1: Average Takeoff, Landing and Parking charges

IATA Code	Fleet Type	Value (m.u.)	IATA Code	Fleet Type	Value (m.u.)	IATA Code	Fleet Type	Value (m.u.)
ACE	NB LND	422,142	ACE	NB PRK	45,034	ACE	WB LND	1511,447
ACE	WB PRK	110,221	AGP	NB LND	422,142	AGP	NB PRK	45,034
AGP	WB LND	1511,447	AGP	WB PRK	110,221	AMS	NB LND	270,271
AMS	NB PRK	101,083	AMS	WB LND	892,548	AMS	WB PRK	333,817
ARN	NB LND	343,459	ARN	NB PRK	109,896	ARN	WB LND	912,815
ARN	WB PRK	362,922	AZS	NB LND	93,472	AZS	NB PRK	17,19
AZS	WB LND	308,685	AZS	WB PRK	56,77	BCN	NB LND	422,142
BCN	NB PRK	45,034	BCN	WB LND	1511,447	BCN	WB PRK	110,221
BIO	NB LND	422,142	BIO	NB PRK	45,034	BIO	WB LND	1511,447
BIO	WB PRK	110,221	BLQ	NB LND	146,248	BLQ	NB PRK	137,645
BLQ	WB LND	482,97	BLQ	WB PRK	454,56	BRU	NB LND	266,916
BRU	NB PRK	1015,13	BRU	WB LND	881,468	BRU	WB PRK	3352,38
BSB	NB LND	275,833	BSB	NB PRK	1321,66	BSB	WB LND	910,915
BSB	WB PRK	4364,668	BUD	NB LND	607,931	BUD	NB PRK	156,284
BUD	WB LND	1609,9	BUD	WB PRK	516,115	CCS	NB LND	292,403
CCS	NB PRK	1169,611	CCS	WB LND	965,635	CCS	WB PRK	3862,538
CDG	NB LND	460,423	CDG	NB PRK	618,398	CDG	WB LND	1779,253
CDG	WB PRK	2042,206	CFE	NB LND	358,014	CFE	NB PRK	272,852
CFE	WB LND	1752,715	CFE	WB PRK	901,07	CMN	NB LND	478,918
CMN	NB PRK	524,793	CMN	WB LND	2081,033	CMN	WB PRK	1733,084
CNF	NB LND	275,833	CNF	NB PRK	1321,66	CNF	WB LND	910,915
CNF	WB PRK	4364,668	CPH	NB LND	621,832	CPH	NB PRK	1572,162
CPH	WB LND	2053,547	CPH	WB PRK	5191,927	DKR	NB LND	0,374
DKR	NB PRK	86,558	DKR	WB LND	1,684	DKR	WB PRK	285,851
DME	NB LND	405,647	DME	NB PRK	40,565	DME	WB LND	1339,613
DME	WB PRK	133,961	EWR	NB LND	617,778	EWR	NB PRK	81,574
EWR	WB LND	2040,157	EWR	WB PRK	396,722	FAO	NB LND	374,939
FAO	NB PRK	102,517	FAO	WB LND	1456,013	FAO	WB PRK	338,553
FCO	NB LND	125,457	FCO	NB PRK	137,645	FCO	WB LND	414,312
FCO	WB PRK	454,56	FNC	NB LND	785,005	FNC	NB PRK	102,517
FNC	WB LND	3049,34	FNC	WB PRK	338,553	FOR	NB LND	275,833
FOR	NB PRK	1321,66	FOR	WB LND	910,915	FOR	WB PRK	4364,668
FRA	NB LND	63,087	FRA	NB PRK	1002	FRA	WB LND	208,34
FRA	WB PRK	1426	FUE	NB LND	422,142	FUE	NB PRK	45,034
FUE	WB LND	1511,447	FUE	WB PRK	110,221	GIG	NB LND	275,833
GIG	NB PRK	1321,66	GIG	WB LND	910,915	GIG	WB PRK	4364,668
GRU	NB LND	275,833	GRU	NB PRK	1321,66	GRU	WB LND	910,915
GRU	WB PRK	4364,668	GVA	NB LND	325,508	GVA	NB PRK	2277,299
GVA	WB LND	1150,232	GVA	WB PRK	7520,582	HAM	NB LND	189,978
HAM	NB PRK	688,224	HAM	WB LND	627,387	HAM	WB PRK	2272,8
HEL	NB LND	407,4	HEL	NB PRK	281,025	HEL	WB LND	1536,41
HEL	WB PRK	928,06	HER	NB LND	99,802	HER	NB PRK	58,499

Continues on the next page...

IATA Code	Fleet	Type	Value (m.u.)	IATA Code	Fleet	Type	Value (m.u.)	IATA Code	Fleet	Type	Value (m.u.)
HER	WB	LND	351,14	HER	WB	PRK	272,736	HOR	NB	LND	263,819
HOR	NB	PRK	2460,401	HOR	WB	LND	1025,128	HOR	WB	PRK	8125,26
JNB	NB	LND	619,312	JNB	NB	PRK	34,115	JNB	WB	LND	2135,866
JNB	WB	PRK	70,99	KGS	NB	LND	99,802	KGS	NB	PRK	58,499
KGS	WB	LND	351,14	KGS	WB	PRK	272,736	LAD	NB	LND	293,839
LAD	NB	PRK	64,329	LAD	WB	LND	1107,327	LAD	WB	PRK	212,44
LCG	NB	LND	379,384	LCG	NB	PRK	45,034	LCG	WB	LND	1359,891
LCG	WB	PRK	110,221	LGW	NB	LND	544,153	LGW	NB	PRK	2729,449
LGW	WB	LND	544,153	LGW	WB	PRK	5983,163	LHR	NB	LND	565,779
LHR	NB	PRK	3416,334	LHR	WB	LND	565,779	LHR	WB	PRK	7614,674
LIN	NB	LND	118,443	LIN	NB	PRK	137,645	LIN	WB	LND	420,502
LIN	WB	PRK	454,56	LIS	NB	LND	374,939	LIS	NB	PRK	102,517
LIS	WB	LND	1456,013	LIS	WB	PRK	338,553	LPA	NB	LND	379,384
LPA	NB	PRK	45,034	LPA	WB	LND	1359,891	LPA	WB	PRK	110,221
LUX	NB	LND	462,401	LUX	NB	PRK	88,896	LUX	WB	LND	1527,038
LUX	WB	PRK	293,57	LYS	NB	LND	304,905	LYS	NB	PRK	449,066
LYS	WB	LND	1541,647	LYS	WB	PRK	1483,002	MAD	NB	LND	422,142
MAD	NB	PRK	45,034	MAD	WB	LND	1511,447	MAD	WB	PRK	110,221
MED	NB	LND	389,105	MED	NB	PRK	77,821	MED	WB	LND	1284,984
MED	WB	PRK	256,997	MPM	NB	LND	353,009	MPM	NB	PRK	194,935
MPM	WB	LND	970,138	MPM	WB	PRK	643,756	MRS	NB	LND	258,063
MRS	NB	PRK	395,729	MRS	WB	LND	1526,101	MRS	WB	PRK	1306,86
MUC	NB	LND	103,95	MUC	NB	PRK	179,225	MUC	WB	LND	343,287
MUC	WB	PRK	591,875	MXP	NB	LND	127,579	MXP	NB	PRK	137,645
MXP	WB	LND	447,795	MXP	WB	PRK	454,56	NAT	NB	LND	275,833
NAT	NB	PRK	1321,66	NAT	WB	LND	910,915	NAT	WB	PRK	4364,668
NCE	NB	LND	505,811	NCE	NB	PRK	430,14	NCE	WB	LND	1446,994
NCE	WB	PRK	1420,5	OPO	NB	LND	374,939	OPO	NB	PRK	102,517
OPO	WB	LND	1456,013	OPO	WB	PRK	338,553	ORY	NB	LND	460,423
ORY	NB	PRK	618,398	ORY	WB	LND	1779,253	ORY	WB	PRK	2042,206
OSL	NB	LND	803,672	OSL	NB	PRK	0	OSL	WB	LND	3420,567
OSL	WB	PRK	0	OXB	NB	LND	338,886	OXB	NB	PRK	584,805
OXB	WB	LND	1404,436	OXB	WB	PRK	1931,269	PDL	NB	LND	263,819
PDL	NB	PRK	102,517	PDL	WB	LND	1025,128	PDL	WB	PRK	338,553
PIX	NB	LND	374,939	PIX	NB	PRK	102,517	PIX	WB	LND	1456,013
PIX	WB	PRK	338,553	PNA	NB	LND	316,168	PNA	NB	PRK	45,034
PNA	WB	LND	1133,948	PNA	WB	PRK	110,221	PRG	NB	LND	517,683
PRG	NB	PRK	935,169	PRG	WB	LND	1215,858	PRG	WB	PRK	3088,316
PUJ	NB	LND	214,879	PUJ	NB	PRK	21,911	PUJ	WB	LND	709,62
PUJ	WB	PRK	72,358	PXO	NB	LND	785,005	PXO	NB	PRK	102,517
PXO	WB	LND	3049,34	PXO	WB	PRK	338,553	RAI	NB	LND	463,868
RAI	NB	PRK	133,594	RAI	WB	LND	1531,885	RAI	WB	PRK	441,183
RAK	NB	LND	333,982	RAK	NB	PRK	45,371	RAK	WB	LND	1644,016
RAK	WB	PRK	149,834	REC	NB	LND	250,492	REC	NB	PRK	1204,699
REC	WB	LND	827,227	REC	WB	PRK	3978,415	SCQ	NB	LND	379,384
SCQ	NB	PRK	45,034	SCQ	WB	LND	1359,891	SCQ	WB	PRK	110,221
SID	NB	LND	463,868	SID	NB	PRK	133,594	SID	WB	LND	1531,885
SID	WB	PRK	441,183	SSA	NB	LND	275,833	SSA	NB	PRK	1321,66
SSA	WB	LND	910,915	SSA	WB	PRK	4364,668	SVQ	NB	LND	422,142
SVQ	NB	PRK	45,034	SVQ	WB	LND	1511,447	SVQ	WB	PRK	110,221
TER	NB	LND	374,939	TER	NB	PRK	102,517	TER	WB	LND	1456,013
TER	WB	PRK	338,553	TLS	NB	LND	292,138	TLS	NB	PRK	387,126

Continues on the next page...

IATA Code	Fleet	Type	Value (m.u.)	IATA Code	Fleet	Type	Value (m.u.)	IATA Code	Fleet	Type	Value (m.u.)
TLS	WB	LND	1390,2	TLS	WB	PRK	1278,45	TMS	NB	LND	194,935
TMS	NB	PRK	1949,351	TMS	WB	LND	643,756	TMS	WB	PRK	6437,564
TUN	NB	LND	645,21	TUN	NB	PRK	275,29	TUN	WB	LND	3551,25
TUN	WB	PRK	909,12	VCE	NB	LND	131,446	VCE	NB	PRK	137,645
VCE	WB	LND	458,265	VCE	WB	PRK	454,56	VRA	NB	LND	248,799
VRA	NB	PRK	49,862	VRA	WB	LND	821,637	VRA	WB	PRK	164,663
WAW	NB	LND	516,238	WAW	NB	PRK	2280,621	WAW	WB	LND	2306,329
WAW	WB	PRK	7531,554	ZAG	NB	LND	810,097	ZAG	NB	PRK	179,225
ZAG	WB	LND	2675,275	ZAG	WB	PRK	591,875	ZRH	NB	LND	406,167
ZRH	NB	PRK	2277,299	ZRH	WB	LND	1449,361	ZRH	WB	PRK	7520,582

Table C.2: City Pairs Distance

Origin	Name	Latitude	Longitude		
Destination	Name	Latitude	Longitude	Dist (nmi)	Dist (km)
ACE	Lanzarote (Spain)	28,94556	-13,60528		
OPO	Porto	41,23547	-8,67796	775,69	1436,58
AGP	Malaga (Spain)	36,675	-4,49917		
LIS	LISBOA	38,77416	-9,13419	253,42	469,34
AMS	Amterdam (Schiphol)	52,30805	4,76416		
LIS	LISBOA	38,77416	-9,13419	996,56	1845,62
AMS	Amterdam (Schiphol)	52,30805	4,76416		
OPO	Porto	41,23547	-8,67796	861,15	1594,85
ARN	Stockholm (Arlanda)	59,65194	17,91859		
LIS	LISBOA	38,77416	-9,13419	1618,19	2996,9
AZS	Saman? (Dominican Republic)	19,27	-69,7375		
LIS	LISBOA	38,77416	-9,13419	3323,7	6155,5
BCN	Barcelona	41,29707	2,07846		
LIS	LISBOA	38,77416	-9,13419	536,42	993,46
BCN	Barcelona	41,29707	2,07846		
OPO	Porto	41,23547	-8,67796	484,81	897,87
BIO	Bilbao	43,30111	-2,91056		
LIS	LISBOA	38,77416	-9,13419	391,08	724,27
BLQ	Bologna (Borgo Panigale)	44,53083	11,29694		
LIS	LISBOA	38,77416	-9,13419	975,68	1806,95
BLQ	Bologna (Borgo Panigale)	44,53083	11,29694		
ZAG	Zagreb (Pleso)	45,74293	16,06877	214,62	397,48
BRU	Brussels	50,90138	4,48457		
LIS	LISBOA	38,77416	-9,13419	927,05	1716,89
BRU	Brussels	50,90138	4,48457		
OPO	Porto	41,23547	-8,67796	795,53	1473,32
BSB	BRASILIA (Pres. J. Kubitschek)	-15,86917	-47,92084		
LIS	LISBOA	38,77416	-9,13419	3935,27	7288,12
BUD	Budapest	47,43931	19,2618		
LIS	LISBOA	38,77416	-9,13419	1338,63	2479,14
BUD	Budapest	47,43931	19,2618		
PRG	Prague	50,10083	14,26	254,12	470,63
CCS	Maiquetia (Simon Bolivar)	10,60118	-66,99124		
FNC	Madeira	32,69424	-16,77808	3062,63	5671,98
CCS	Maiquetia (Simon Bolivar)	10,60118	-66,99124		

Continues on the next page...

Origin	Name	Latitude	Longitude		
Destination	Name	Latitude	Longitude	Dist (nmi)	Dist (km)
LIS	LISBOA	38,77416	-9,13419	3508,31	6497,39
CCS	Maiquetia (Simon Bolivar)	10,60118	-66,99124		
OPO	Porto	41,23547	-8,67796	3562,03	6596,89
CDG	Paris (Charles de Gaule France)	49,00973	2,5478		
BRU	Brussels	50,90138	4,48457	135,9	251,69
CDG	Paris (Charles de Gaule France)	49,00973	2,5478		
LIS	LISBOA	38,77416	-9,13419	793,29	1469,17
CDG	Paris (Charles de Gaule France)	49,00973	2,5478		
LUX	Luxembourg	49,62325	6,20431	147,66	273,47
CDG	Paris (Charles de Gaule France)	49,00973	2,5478		
OPO	Porto	41,23547	-8,67796	664,59	1230,82
CMN	Mohammed V (Marocco)	33,36722	-7,58972		
LIS	LISBOA	38,77416	-9,13419	332,93	616,6
CNF	Belo Horizonte (Tancredo Neves)	-19,62446	-43,97196		
LIS	LISBOA	38,77416	-9,13419	4013,79	7433,54
CPH	Copenhagen Denmark	55,6179	12,65597		
ARN	Stockholm (Arlanda)	59,65194	17,91859	295,03	546,39
CPH	Copenhagen Denmark	55,6179	12,65597		
LIS	LISBOA	38,77416	-9,13419	1334,08	2470,71
DKR	Dakar (Leopold S. Senghor)	14,74389	-17,47946		
LIS	LISBOA	38,77416	-9,13419	1507,8	2792,44
DME	MOSCOW/DOMODEDOVO	55,40833	37,90833		
LIS	LISBOA	38,77416	-9,13419	2111,9	3911,24
EWR	Newark	40,69248	-74,16868		
LIS	LISBOA	38,77416	-9,13419	2932	5430,07
EWR	Newark	40,69248	-74,16868		
OPO	Porto	41,23547	-8,67796	2893,05	5357,93
FAO	Faro	37,01444	-7,96584		
LIS	LISBOA	38,77416	-9,13419	119,19	220,75
FAO	Faro	37,01444	-7,96584		
OPO	Porto	41,23547	-8,67796	255,42	473,03
FCO	Roma (Fiumicino)	41,80028	12,23888		
LIS	LISBOA	38,77416	-9,13419	992,22	1837,6
FCO	Roma (Fiumicino)	41,80028	12,23888		
OPO	Porto	41,23547	-8,67796	937,98	1737,14
FNC	Madeira	32,69424	-16,77808		
CCS	Maiquetia (Simon Bolivar)	10,60118	-66,99124	3062,63	5671,98
FNC	Madeira	32,69424	-16,77808		
LGW	London (Gatwick)	51,14805	-0,19029	1325,27	2454,39
FNC	Madeira	32,69424	-16,77808		
LIS	LISBOA	38,77416	-9,13419	520,84	964,59
FNC	Madeira	32,69424	-16,77808		
MAD	Madrid (Barajas)	40,47222	-3,56096	788,04	1459,44
FNC	Madeira	32,69424	-16,77808		
OPO	Porto	41,23547	-8,67796	642,32	1189,58
FOR	Fortaleza (Pinto Martins)	-3,77612	-38,53251		
LIS	LISBOA	38,77416	-9,13419	3028,08	5608,01
FRA	Frankfurt (Main)	50,03331	8,57045		
LIS	LISBOA	38,77416	-9,13419	1011,37	1873,06
FUE	Fuerteventura (Spain)	28,45278	-13,86389		
OPO	Porto	41,23547	-8,67796	807,94	1496,3

Continues on the next page...

Origin	Name	Latitude	Longitude		
Destination	Name	Latitude	Longitude	Dist (nmi)	Dist (km)
GIG	Rio De Janeiro (Galeao)	-22,81002	-43,25056		
LIS	LISBOA	38,77416	-9,13419	4162,73	7709,38
GIG	Rio De Janeiro (Galeao)	-22,81002	-43,25056		
OPO	Porto	41,23547	-8,67796	4297,47	7958,91
GRU	Sao Paulo (Guarulhos)	-23,43558	-46,47307		
LIS	LISBOA	38,77416	-9,13419	4281,52	7929,38
GRU	Sao Paulo (Guarulhos)	-23,43558	-46,47307		
OPO	Porto	41,23547	-8,67796	4413,41	8173,64
GVA	Geneva (Cointrin)	46,23832	6,01094		
FAO	Faro	37,01444	-7,96584	833,97	1544,51
GVA	Geneva (Cointrin)	46,23832	6,01094		
LIS	LISBOA	38,77416	-9,13419	803,68	1488,42
GVA	Geneva (Cointrin)	46,23832	6,01094		
OPO	Porto	41,23547	-8,67796	702,54	1301,11
GVA	Geneva (Cointrin)	46,23832	6,01094		
ZRH	Zurich	47,45832	8,54805	127,26	235,68
HAM	Hamburg	53,63038	9,98823		
LIS	LISBOA	38,77416	-9,13419	1186,15	2196,76
HEL	Helsinki (Vantaa)	60,31722	24,96332		
LIS	LISBOA	38,77416	-9,13419	1815,44	3362,19
HER	Heraklion (Greece)	35,33583	25,17361		
LIS	LISBOA	38,77416	-9,13419	1645,98	3048,35
HOR	Horta	38,51989	-28,71639		
LIS	LISBOA	38,77416	-9,13419	916	1696,44
JNB	Johannesburg (O. R. Tambo Intl)	-26,13371	28,24231		
LIS	LISBOA	38,77416	-9,13419	4423,11	8191,6
JNB	Johannesburg (O. R. Tambo Intl)	-26,13371	28,24231		
MPM	Maputo	-25,92085	32,5726	233,81	433,01
KGS	Kos Island (Greece)	36,79334	27,09167		
HER	Heraklion (Greece)	35,33583	25,17361	127,67	236,45
LAD	Luanda	-8,85771	13,22833		
LIS	LISBOA	38,77416	-9,13419	3118,7	5775,83
LCG	A Coru?a (Spain)	43,30194	-8,37722		
LIS	LISBOA	38,77416	-9,13419	273,81	507,1
LGW	London (Gatwick)	51,14805	-0,19029		
FNC	Madeira	32,69424	-16,77808	1325,27	2454,39
LGW	London (Gatwick)	51,14805	-0,19029		
LIS	LISBOA	38,77416	-9,13419	832,47	1541,73
LGW	London (Gatwick)	51,14805	-0,19029		
OPO	Porto	41,23547	-8,67796	690,34	1278,51
LHR	London (Heathrow)	51,47749	-0,46141		
LIS	LISBOA	38,77416	-9,13419	844,62	1564,24
LHR	London (Heathrow)	51,47749	-0,46141		
OPO	Porto	41,23547	-8,67796	701,41	1299,01
LIN	MilaN (Linate)	45,44943	9,27832		
LIS	LISBOA	38,77416	-9,13419	909,3	1684,03
LIS	LISBOA	38,77416	-9,13419		
ACE	Lanzarote (Spain)	28,94556	-13,60528	630,16	1167,05
LIS	LISBOA	38,77416	-9,13419		
AGP	Malaga (Spain)	36,675	-4,49917	253,42	469,34
LIS	LISBOA	38,77416	-9,13419		

Continues on the next page...

Origin	Name	Latitude	Longitude		
Destination	Name	Latitude	Longitude	Dist (nmi)	Dist (km)
AMS	Amterdam (Schiphol)	52,30805	4,76416	996,56	1845,62
LIS	LISBOA	38,77416	-9,13419		
ARN	Stockholm (Arlanda)	59,65194	17,91859	1618,19	2996,9
LIS	LISBOA	38,77416	-9,13419		
AZS	Saman? (Dominican Republic)	19,27	-69,7375	3323,7	6155,5
LIS	LISBOA	38,77416	-9,13419		
BCN	Barcelona	41,29707	2,07846	536,42	993,46
LIS	LISBOA	38,77416	-9,13419		
BIO	Bilbao	43,30111	-2,91056	391,08	724,27
LIS	LISBOA	38,77416	-9,13419		
BLQ	Bologna (Borgo Panigale)	44,53083	11,29694	975,68	1806,95
LIS	LISBOA	38,77416	-9,13419		
BRU	Brussels	50,90138	4,48457	927,05	1716,89
LIS	LISBOA	38,77416	-9,13419		
BSB	BRASILIA (Pres. J. Kubitschek)	-15,86917	-47,92084	3935,27	7288,12
LIS	LISBOA	38,77416	-9,13419		
BUD	Budapest	47,43931	19,2618	1338,63	2479,14
LIS	LISBOA	38,77416	-9,13419		
CCS	Maiquetia (Simon Bolivar)	10,60118	-66,99124	3508,31	6497,39
LIS	LISBOA	38,77416	-9,13419		
CDG	Paris (Charles de Gaule France)	49,00973	2,5478	793,29	1469,17
LIS	LISBOA	38,77416	-9,13419		
CMN	Mohammed V (Marocco)	33,36722	-7,58972	332,93	616,6
LIS	LISBOA	38,77416	-9,13419		
CNF	Belo Horizonte (Tancredo Neves)	-19,62446	-43,97196	4013,79	7433,54
LIS	LISBOA	38,77416	-9,13419		
CPH	Copenhagen Denmark	55,6179	12,65597	1334,08	2470,71
LIS	LISBOA	38,77416	-9,13419		
DKR	Dakar (Leopold S. Senghor)	14,74389	-17,47946	1507,8	2792,44
LIS	LISBOA	38,77416	-9,13419		
DME	MOSCOW/DOMODEDOVO	55,40833	37,90833	2111,9	3911,24
LIS	LISBOA	38,77416	-9,13419		
EWR	Newark	40,69248	-74,16868	2932	5430,07
LIS	LISBOA	38,77416	-9,13419		
FAO	Faro	37,01444	-7,96584	119,19	220,75
LIS	LISBOA	38,77416	-9,13419		
FCO	Roma (Fiumicino)	41,80028	12,23888	992,22	1837,6
LIS	LISBOA	38,77416	-9,13419		
FNC	Madeira	32,69424	-16,77808	520,84	964,59
LIS	LISBOA	38,77416	-9,13419		
FOR	Fortaleza (Pinto Martins)	-3,77612	-38,53251	3028,08	5608,01
LIS	LISBOA	38,77416	-9,13419		
FRA	Frankfurt (Main)	50,03331	8,57045	1011,37	1873,06
LIS	LISBOA	38,77416	-9,13419		
FUE	Fuerteventura (Spain)	28,45278	-13,86389	662,57	1227,08
LIS	LISBOA	38,77416	-9,13419		
GIG	Rio De Janeiro (Galeao)	-22,81002	-43,25056	4162,73	7709,38
LIS	LISBOA	38,77416	-9,13419		
GRU	Sao Paulo (Guarulhos)	-23,43558	-46,47307	4281,52	7929,38
LIS	LISBOA	38,77416	-9,13419		
GVA	Geneva (Cointrin)	46,23832	6,01094	803,68	1488,42

Continues on the next page...

Origin	Name	Latitude	Longitude		
Destination	Name	Latitude	Longitude	Dist (nmi)	Dist (km)
LIS	LISBOA	38,77416	-9,13419		
HAM	Hamburg	53,63038	9,98823	1186,15	2196,76
LIS	LISBOA	38,77416	-9,13419		
HEL	Helsinki (Vantaa)	60,31722	24,96332	1815,44	3362,19
LIS	LISBOA	38,77416	-9,13419		
HOR	Horta	38,51989	-28,71639	916	1696,44
LIS	LISBOA	38,77416	-9,13419		
JNB	Johannesburg (O. R. Tambo Intl)	-26,13371	28,24231	4423,11	8191,6
LIS	LISBOA	38,77416	-9,13419		
KGS	Kos Island (Greece)	36,79334	27,09167	1710,72	3168,25
LIS	LISBOA	38,77416	-9,13419		
LAD	Luanda	-8,85771	13,22833	3118,7	5775,83
LIS	LISBOA	38,77416	-9,13419		
LCG	A Coru?a (Spain)	43,30194	-8,37722	273,81	507,1
LIS	LISBOA	38,77416	-9,13419		
LGW	London (Gatwick)	51,14805	-0,19029	832,47	1541,73
LIS	LISBOA	38,77416	-9,13419		
LHR	London (Heathrow)	51,47749	-0,46141	844,62	1564,24
LIS	LISBOA	38,77416	-9,13419		
LIN	MilaN (Linate)	45,44943	9,27832	909,3	1684,03
LIS	LISBOA	38,77416	-9,13419		
LPA	Gran Canaria (Spain)	27,93194	-15,38667	721,58	1336,37
LIS	LISBOA	38,77416	-9,13419		
LUX	Luxembourg	49,62325	6,20431	923,47	1710,26
LIS	LISBOA	38,77416	-9,13419		
LYS	Lyon (St Exupery)	45,72563	5,08109	754,94	1398,15
LIS	LISBOA	38,77416	-9,13419		
MAD	Madrid (Barajas)	40,47222	-3,56096	276,92	512,86
LIS	LISBOA	38,77416	-9,13419		
MPM	Maputo	-25,92085	32,5726	4532,32	8393,86
LIS	LISBOA	38,77416	-9,13419		
MRS	Marseille (Provence)	43,43666	5,21499	705,26	1306,14
LIS	LISBOA	38,77416	-9,13419		
MUC	Munich	48,35377	11,78608	1070,55	1982,66
LIS	LISBOA	38,77416	-9,13419		
MXP	Milan (Malpensa)	45,63	8,72304	891,2	1650,51
LIS	LISBOA	38,77416	-9,13419		
NAT	Natal	-5,90835	-35,24918	3048,81	5646,39
LIS	LISBOA	38,77416	-9,13419		
NCE	Nice (Cote D Azur)	43,6654	7,21497	792,38	1467,48
LIS	LISBOA	38,77416	-9,13419		
OPO	Porto	41,23547	-8,67796	149,16	276,24
LIS	LISBOA	38,77416	-9,13419		
ORY	Paris (Orly)	48,72327	2,37957	776,29	1437,7
LIS	LISBOA	38,77416	-9,13419		
OSL	Oslo (Gardemoen)	60,20276	11,08388	1494,34	2767,52
LIS	LISBOA	38,77416	-9,13419		
OXB	Bissau (Osvaldo Vieira)	11,8886	-15,65556	1650,27	3056,29
LIS	LISBOA	38,77416	-9,13419		
PDL	Ponta Delgada	37,74196	-25,69766	781,75	1447,8
LIS	LISBOA	38,77416	-9,13419		

Continues on the next page...

Origin	Name	Latitude	Longitude		
Destination	Name	Latitude	Longitude	Dist (nmi)	Dist (km)
PIX	Pico	38,55433	-28,44135	902,94	1672,24
LIS	LISBOA	38,77416	-9,13419		
PNA	Pamplona (Spain)	42,77	-1,647	415,93	770,31
LIS	LISBOA	38,77416	-9,13419		
PRG	Prague	50,10083	14,26	1202,79	2227,57
LIS	LISBOA	38,77416	-9,13419		
PUJ	Punta Cana (Dominican Republic)	18,56667	-68,35194	3282,96	6080,05
LIS	LISBOA	38,77416	-9,13419		
PXO	Porto Santo	33,07082	-16,34974	489,59	906,72
LIS	LISBOA	38,77416	-9,13419		
RAI	Praia	14,94111	-23,48471	1618,36	2997,2
LIS	LISBOA	38,77416	-9,13419		
RAK	Marrakech (Marocco)	31,60694	-8,03639	433,38	802,62
LIS	LISBOA	38,77416	-9,13419		
REC	Recife (Guararapes)	-8,12641	-34,92279	3160,22	5852,73
LIS	LISBOA	38,77416	-9,13419		
SID	Sal (Amilcar Cabral)	16,74194	-22,94889	1507,55	2791,99
LIS	LISBOA	38,77416	-9,13419		
SSA	Salvador (Dep. Luis Magalhaes)	-12,90861	-38,3225	3505,87	6492,88
LIS	LISBOA	38,77416	-9,13419		
SVQ	Seville (Spain)	37,41806	-5,89889	173,07	320,52
LIS	LISBOA	38,77416	-9,13419		
TER	Lajes	38,76193	-27,0909	838,68	1553,24
LIS	LISBOA	38,77416	-9,13419		
TLS	Toulouse (Blagnac)	43,63512	1,36784	555,96	1029,65
LIS	LISBOA	38,77416	-9,13419		
TUN	Tunis - Carthage (Tunisia)	36,85111	10,22722	923,22	1709,8
LIS	LISBOA	38,77416	-9,13419		
VCE	Venice (Tessera)	45,50526	12,35194	1033,65	1914,32
LIS	LISBOA	38,77416	-9,13419		
VRA	Varadero (Cuba)	23,03444	-81,43528	3744,52	6934,85
LIS	LISBOA	38,77416	-9,13419		
WAW	Warszawa - Ocecie Poland	52,016	20,967	1480,38	2741,66
LIS	LISBOA	38,77416	-9,13419		
ZAG	Zagreb (Pleso)	45,74293	16,06877	1188,63	2201,34
LIS	LISBOA	38,77416	-9,13419		
ZRH	Zurich	47,45832	8,54805	929,98	1722,32
LPA	Gran Canaria (Spain)	27,93194	-15,38667		
OPO	Porto	41,23547	-8,67796	863,55	1599,3
LUX	Luxembourg	49,62325	6,20431		
LIS	LISBOA	38,77416	-9,13419	923,47	1710,26
LUX	Luxembourg	49,62325	6,20431		
OPO	Porto	41,23547	-8,67796	801,23	1483,89
LYS	Lyon (St Exupery)	45,72563	5,08109		
LIS	LISBOA	38,77416	-9,13419	754,94	1398,15
MAD	Madrid (Barajas)	40,47222	-3,56096		
FNC	Madeira	32,69424	-16,77808	788,04	1459,44
MAD	Madrid (Barajas)	40,47222	-3,56096		
LIS	LISBOA	38,77416	-9,13419	276,92	512,86
MAD	Madrid (Barajas)	40,47222	-3,56096		
OPO	Porto	41,23547	-8,67796	236,66	438,29

Continues on the next page...

Origin	Name	Latitude	Longitude		
Destination	Name	Latitude	Longitude	Dist (nmi)	Dist (km)
MED	Medina (Saudi Arabia)	24,55333	39,705		
LIS	LISBOA	38,77416	-9,13419	2602,07	4819,04
MPM	Maputo	-25,92085	32,5726		
JNB	Johannesburg (O. R. Tambo Intl)	-26,13371	28,24231	233,81	433,01
MPM	Maputo	-25,92085	32,5726		
LIS	LISBOA	38,77416	-9,13419	4532,32	8393,86
MRS	Marseille (Provence)	43,43666	5,21499		
LIS	LISBOA	38,77416	-9,13419	705,26	1306,14
MUC	Munich	48,35377	11,78608		
FAO	Faro	37,01444	-7,96584	1099,8	2036,83
MUC	Munich	48,35377	11,78608		
LIS	LISBOA	38,77416	-9,13419	1070,55	1982,66
MXP	Milan (Malpensa)	45,63	8,72304		
LIS	LISBOA	38,77416	-9,13419	891,2	1650,51
MXP	Milan (Malpensa)	45,63	8,72304		
OPO	Porto	41,23547	-8,67796	800,76	1483,01
NAT	Natal	-5,90835	-35,24918		
LIS	LISBOA	38,77416	-9,13419	3048,81	5646,39
NCE	Nice (Cote D Azur)	43,6654	7,21497		
LIS	LISBOA	38,77416	-9,13419	792,38	1467,48
OPO	Porto	41,23547	-8,67796		
AMS	Amterdam (Schiphol)	52,30805	4,76416	861,15	1594,85
OPO	Porto	41,23547	-8,67796		
BCN	Barcelona	41,29707	2,07846	484,81	897,87
OPO	Porto	41,23547	-8,67796		
BRU	Brussels	50,90138	4,48457	795,53	1473,32
OPO	Porto	41,23547	-8,67796		
CCS	Maiquetia (Simon Bolivar)	10,60118	-66,99124	3562,03	6596,89
OPO	Porto	41,23547	-8,67796		
CDG	Paris (Charles de Gaule France)	49,00973	2,5478	664,59	1230,82
OPO	Porto	41,23547	-8,67796		
EWR	Newark	40,69248	-74,16868	2893,05	5357,93
OPO	Porto	41,23547	-8,67796		
FAO	Faro	37,01444	-7,96584	255,42	473,03
OPO	Porto	41,23547	-8,67796		
FCO	Roma (Fiumicino)	41,80028	12,23888	937,98	1737,14
OPO	Porto	41,23547	-8,67796		
FNC	Madeira	32,69424	-16,77808	642,32	1189,58
OPO	Porto	41,23547	-8,67796		
GIG	Rio De Janeiro (Galeao)	-22,81002	-43,25056	4297,47	7958,91
OPO	Porto	41,23547	-8,67796		
GRU	Sao Paulo (Guarulhos)	-23,43558	-46,47307	4413,41	8173,64
OPO	Porto	41,23547	-8,67796		
GVA	Geneva (Cointrin)	46,23832	6,01094	702,54	1301,11
OPO	Porto	41,23547	-8,67796		
LGW	London (Gatwick)	51,14805	-0,19029	690,34	1278,51
OPO	Porto	41,23547	-8,67796		
LHR	London (Heathrow)	51,47749	-0,46141	701,41	1299,01
OPO	Porto	41,23547	-8,67796		
LIS	LISBOA	38,77416	-9,13419	149,16	276,24
OPO	Porto	41,23547	-8,67796		

Continues on the next page...

Origin	Name	Latitude	Longitude		
Destination	Name	Latitude	Longitude	Dist (nmi)	Dist (km)
LUX	Luxembourg	49,62325	6,20431	801,23	1483,89
OPO	Porto	41,23547	-8,67796		
MAD	Madrid (Barajas)	40,47222	-3,56096	236,66	438,29
OPO	Porto	41,23547	-8,67796		
MXP	Milan (Malpensa)	45,63	8,72304	800,76	1483,01
OPO	Porto	41,23547	-8,67796		
ORY	Paris (Orly)	48,72327	2,37957	648,45	1200,93
OPO	Porto	41,23547	-8,67796		
PXO	Porto Santo	33,07082	-16,34974	611,46	1132,43
OPO	Porto	41,23547	-8,67796		
SCQ	Santiago de Compostela (Spain)	42,89639	-8,41528	100,34	185,83
OPO	Porto	41,23547	-8,67796		
TUN	Tunis - Carthage (Tunisia)	36,85111	10,22722	917,28	1698,8
OPO	Porto	41,23547	-8,67796		
ZRH	Zurich	47,45832	8,54805	825,6	1529,01
ORY	Paris (Orly)	48,72327	2,37957		
LIS	LISBOA	38,77416	-9,13419	776,29	1437,7
ORY	Paris (Orly)	48,72327	2,37957		
OPO	Porto	41,23547	-8,67796	648,45	1200,93
OSL	Oslo (Gardemoen)	60,20276	11,08388		
LIS	LISBOA	38,77416	-9,13419	1494,34	2767,52
OXB	Bissau (Osvaldo Vieira)	11,8886	-15,65556		
LIS	LISBOA	38,77416	-9,13419	1650,27	3056,29
PDL	Ponta Delgada	37,74196	-25,69766		
LIS	LISBOA	38,77416	-9,13419	781,75	1447,8
PIX	Pico	38,55433	-28,44135		
TER	Lajes	38,76193	-27,0909	64,49	119,43
PNA	Pamplona (Spain)	42,77	-1,647		
LIS	LISBOA	38,77416	-9,13419	415,93	770,31
PRG	Prague	50,10083	14,26		
BUD	Budapest	47,43931	19,2618	254,12	470,63
PRG	Prague	50,10083	14,26		
LIS	LISBOA	38,77416	-9,13419	1202,79	2227,57
PUJ	Punta Cana (Dominican Republic)	18,56667	-68,35194		
LIS	LISBOA	38,77416	-9,13419	3282,96	6080,05
PXO	Porto Santo	33,07082	-16,34974		
FNC	Madeira	32,69424	-16,77808	31,25	57,87
PXO	Porto Santo	33,07082	-16,34974		
LIS	LISBOA	38,77416	-9,13419	489,59	906,72
PXO	Porto Santo	33,07082	-16,34974		
OPO	Porto	41,23547	-8,67796	611,46	1132,43
RAI	Praia	14,94111	-23,48471		
LIS	LISBOA	38,77416	-9,13419	1618,36	2997,2
RAK	Marrakech (Marocco)	31,60694	-8,03639		
MED	Medina (Saudi Arabia)	24,55333	39,705	2542,67	4709,02
REC	Recife (Guararapes)	-8,12641	-34,92279		
LIS	LISBOA	38,77416	-9,13419	3160,22	5852,73
SCQ	Santiago de Compostela (Spain)	42,89639	-8,41528		
LGW	London (Gatwick)	51,14805	-0,19029	597,77	1107,07
SID	Sal (Amilcar Cabral)	16,74194	-22,94889		
LIS	LISBOA	38,77416	-9,13419	1507,55	2791,99

Continues on the next page...

Origin	Name	Latitude	Longitude		
Destination	Name	Latitude	Longitude	Dist (nmi)	Dist (km)
SSA	Salvador (Dep. Luis Magalhaes)	-12,90861	-38,3225		
LIS	LISBOA	38,77416	-9,13419	3505,87	6492,88
SVQ	Seville (Spain)	37,41806	-5,89889		
LIS	LISBOA	38,77416	-9,13419	173,07	320,52
TER	Lajes	38,76193	-27,0909		
LIS	LISBOA	38,77416	-9,13419	838,68	1553,24
TLS	Toulouse (Blagnac)	43,63512	1,36784		
LIS	LISBOA	38,77416	-9,13419	555,96	1029,65
TUN	Tunis - Carthage (Tunisia)	36,85111	10,22722		
LIS	LISBOA	38,77416	-9,13419	923,22	1709,8
TUN	Tunis - Carthage (Tunisia)	36,85111	10,22722		
OPO	Porto	41,23547	-8,67796	917,28	1698,8
VCE	Venice (Tessera)	45,50526	12,35194		
LIS	LISBOA	38,77416	-9,13419	1033,65	1914,32
VRA	Varadero (Cuba)	23,03444	-81,43528		
LIS	LISBOA	38,77416	-9,13419	3744,52	6934,85
WAW	Warszawa - Ocecie Poland	52,016	20,967		
LIS	LISBOA	38,77416	-9,13419	1480,38	2741,66
ZAG	Zagreb (Pleso)	45,74293	16,06877		
BLQ	Bologna (Borgo Panigale)	44,53083	11,29694	214,62	397,48
ZAG	Zagreb (Pleso)	45,74293	16,06877		
LIS	LISBOA	38,77416	-9,13419	1188,63	2201,34
ZRH	Zurich	47,45832	8,54805		
FAO	Faro	37,01444	-7,96584	961,22	1780,19
ZRH	Zurich	47,45832	8,54805		
LIS	LISBOA	38,77416	-9,13419	929,98	1722,32
ZRH	Zurich	47,45832	8,54805		
OPO	Porto	41,23547	-8,67796	825,6	1529,01

Table C.3 Average ATC, Maintenance, Fuel and Handling Costs by Aircraft Model

Aircraft Model	Description	Fleet	Pax	MTOW	ATC Cost (per nmi)	Maint Cost (per min)	Fuel Cost (per min)	Airp Hand (per a/c)
A310	AIRBUS A310-304	WB	202	138600	2,24	16,17	22,99	1386
A319	AIRBUS A319-111	NB	132	68000	2,38	10,5	15,34	950
A320	AIRBUS A320-211	NB	156	73500	2,88	10,33	16,52	950
A321	AIRBUS A321-211	NB	194	83000	2,56	12	19,23	1000
A330	AIRBUS A330-200	WB	230	230000	2,24	16,17	37,11	2250
A340	AIRBUS A340-312	WB	274	257000	2,75	25	43,78	2520
BEH	BEECHCRAFT	NB	19	7688	2,38	6,17	6,35	100
ER4	EMBRAER RJ145	NB	45	20990	2,38	7,67	8,29	350
F100	FOKKER 100	NB	99	43090	2,38	9	12,9	580

Table C.4 Crew member DHC (Extra-Crew) Costs

Airline Company	Cost per flight (m.u.)
8X	30
NI	30
S4	30
TM	30
TP	0
UA	30

Table C.5 Hotel Costs per Night for Crew members and Disrupted Passengers

Hotel Code	Airport	Crewmember Cost per night (m.u.)	Passenger Cost per night (m.u.)
AMSHTL	AMS	103,4	113,74
BCNMEL	BCN	83,5	91,85
BRUHIL	BRU	63,5	69,85
BSBALV	BSB	63,45	69,8
CCSMEL	CCS	64,07	70,48
CDGHTL	CDG	75	82,5
CNFHTL	CNF	57,68	63,45
CPHRDS	CPH	123,63	135,99
FAOEVA	FAO	30,67	33,73
FCOSHT	FCO	66,75	73,43
FNCHTL	FNC	47	51,7
FORHT3	FOR	42,3	46,53
FRAHTL	FRA	63	69,3
GVANH	GVA	69,49	76,44
HAMHTL	HAM	68	74,8
HELHTL	HEL	58	63,8
HILORY	ORY	60	66
JNBRAD	JNB	148,21	163,03
LHRHTL	LHR	64,58	71,04
LINSHT	LIN	79	86,9
LISOLI	LIS	46,5	51,15
MADALA	MAD	59,6	65,56
MPMROV	MPM	57,78	63,56
MUCHTL	MUC	59	64,9
NATHTL	NAT	58,45	64,3
NYCHTL	EWR	62,44	68,69
OPOTIR	OPO	42	46,2
RECPAL	REC	40,28	44,31
RIOINT	GIG	49,45	54,39
SAOHTL	GRU	39,72	43,69
SSAHTL	SSA	54,9	60,39
ZRHHTL	ZRH	84,71	93,18

Table C.6 Monthly and Hourly Salary and Perdiem Values for each Crew Salary Rank

Salary Rank	Monthly Salary (m.u.)	Hourly Salary (m.u.)	Perdiem (m.u.)
CAB0	726	18	31
CAB1	924	23	66
CAB2	1238	31	67
CAB3	1540	39	68
CAB4	1712	43	69
CAB5	1795	45	70
CABi	582	15	31
CC1	2029	51	71
CC2	2087	52	72
CC3	2134	53	73
CTE	5886	130	111
OP1	4475	99	86
OP2	3856	85	77
SC1	2316	58	74
SC2	2441	61	75
SC3	2505	63	76

Appendix D
Delay Codes

Whenever a flight suffers an anomaly on the ground, airline companies use a set of proprietary delay codes to signal what happened wrong. As an attempt to help airlines standardize the reasons behind commercial flight late departures, IATA created a set of delay codes.

This appendix[1] is intended to gather all the delay codes, both IATA and TAP's, for easier reference. Given their general purpose, IATA codes are grouped into related operational fields and offer a more detailed textual description.

These tables are presented in Section D.1 and D.2, respectively. Additionally, in Section D.3 we also include the IATA and TAP delay codes classification, according to the TAP problem classification presented in Section 4.5.

D.1 IATA Numeric Delay Codes and Description

Table D.1: IATA numeric delay codes.

code	label	description
Others		
6	NO GATE/STAND AVAILABLE	Due to own airline activity
9	SCHEDULED GROUND TIME	Planned turnaround time less than declared minimum
Passenger and Baggage		
11	LATE CHECK-IN	Check-in reopened for late passengers
12	LATE CHECK-IN	Check-in not completed by flight closure time
13	CHECK-IN ERROR	Error with passenger or baggage details
14	OVER-SALES	Booking errors not resolved at check-in
15	BOARDING	Discrepancies and paging, missing checked in passengers
16	COMMERCIAL PUBLICITY OR PASSENGER CONVENIENCE	Local decision to delay for VIP or press, delay due to offload of passengers following family bereavement
17	CATERING ORDER	Late or incorrect order given to supplier
18	BAGGAGE PROCESSING	Late or incorrectly sorted baggage
Cargo and Mail		
21	DOCUMENTATION	Late or incorrect documentation for booked cargo
22	LATE POSITIONING	Late delivery of booked cargo to airport/aircraft
23	LATE ACCEPTANCE	Acceptance of cargo after deadline
24	INADEQUATE PACKING	Repackaging and/or re-labeling of booked cargo
25	OVER-SALES	Booked load in excess of saleable load capacity (weight or volume), resulting in reloading or off-load
27	DOCUMENTATION/PACKING	(Mail Only) Incomplete and/or inaccurate documentation
28	LATE POSITIONING	(Mail Only) Late delivery of mail to airport/aircraft

[1] This information also appears in Nuno Machado MsC Thesis(Machado, 2010)

29	LATE ACCEPTANCE	(Mail Only) Acceptance of mail after deadline

Aircraft and Ramp Handling		
31	LATE/INACCURATE AIRCRAFT DOCUMENTATION	Late or inaccurate mass and balance documentation, general declaration, passenger manifest
32	LOADING/UNLOADING	Bulky items, special load, lack loading staff
33	LOADING EQUIPMENT	Lack of and/or breakdown, lack of operating staff
34	SERVICING EQUIPMENT	Lack of and/or breakdown, lack of operating staff
35	AIRCRAFT CLEANING	Late completion of aircraft cleaning
36	FUELLING/DEFUELLING	Late delivery of fuel, excludes late request
37	CATERING	Late and/or incomplete delivery, late loading
38	ULD	Lack of and/or unserviceable ULDs or pallets
39	TECHNICAL EQUIPMENT	Lack and/or breakdown, lack of operating staff, includes GPU, air start, push-back tug, de-icing

Technical and Aircraft Equipment		
41	TECHNICAL DEFECTS	Aircraft defects including items covered by MEL
42	SCHEDULED MAINTENANCE	Late release from maintenance
43	NON-SCHEDULED MAINTENANCE	Special checks and/or additional works beyond normal maintenance schedule
44	SPARES AND MAINTENANCE	Lack of spares, lack of and/or breakdown of specialist equipment required for defect rectification
45	AOG SPARES	Awaiting AOG spare(s) to be carried to another station
46	AIRCRAFT CHANGE	For technical reasons, e.g. a prolonged technical delay
47	STANDBY AIRCRAFT	Standby aircraft unavailable for technical reasons

Damage to Aircraft		
51	DAMAGE DURING FLIGHT OPERATIONS	Bird or lightning strike, turbulence, heavy or overweight landing, collisions during taxiing
52	DAMAGE DURING GROUND OPERATIONS	Collisions (other than taxiing), loading/offloading damage, towing, contamination, extreme weather conditions

EDP/Automated Equipment Failure		
55	DEPARTURE CONTROL	Failure of automated systems, including check-in, load control systems producing mass and balance
56	CARGO PREPARATION DOCUMENTATION	Failure of documentation and/or load control systems covering cargo
57	FLIGHT PLANS	Failure of automated flight plan systems

Flight Operations and Crewing		
61	FLIGHT PLAN	Late completion of or change to flight plan
62	OPERATIONAL REQUIREMENT	Late alteration to fuel or payload
63	LATE CREW BOARDING OR DEPARTURE PROCEDURES	Late flight deck, or entire crew, other than standby; late completion of flight deck crew checks
64	FLIGHT DECK CREW SHORTAGE	Sickness, awaiting standby, flight time limitations, valid visa, health documents, etc.
65	FLIGHT DECK CREW SPECIAL REQUEST	Requests not within operational requirements
66	LATE CABIN CREW BOARDING OR DEPARTURE PROCEDURES	Late cabin crew other than standby, late completion of cabin crew checks
67	CABIN CREW SHORTAGE	Sickness, awaiting standby, flight time limitations, valid visa, health documents

68	CABIN CREW ERROR OR SPECIAL REQUEST	Requests not within operational requirements
69	CAPTAIN REQUEST FOR SECURITY CHECK	Extraordinary requests outside mandatory requirements

Weather		
71	DEPARTURE STATION	Below operating limits
72	DESTINATION STATION	Below operating limits
73	EN-ROUTE OR ALTERNATE	Below operating limits
75	DE-ICING OF AIRCRAFT	Removal of ice and/or snow; excludes equipment
76	REMOVAL OF SNOW, ICE OR SAND FROM AIRPORT	Runway, taxiway conditions
77	GROUND HANDLING IMPAIRED BY ADVERSE CONDITIONS	High winds, heavy rain, blizzards, monsoons etc.

Air Traffic Flow Management Restrictions		
81	ATFM DUE TO ATC EN-ROUTE DEMAND/CAPACITY	Standard demand/capacity problems
82	ATFM DUE TO ATC STAFF OR EQUIPMENT EN ROUTE	Reduced capacity caused by industrial action or staff shortage, equipment failure, military exercise or extraordinary demand due to capacity reduction in neighboring area
83	ATFM DUE TO RESTRICTION AT DESTINATION AIRPORT	Airport and/or runway closed due to obstruction, industrial action, staff shortage, political unrest, noise abatement, night curfew, special flights
84	ATFM DUE TO WEATHER AT DESTINATION	Airport and/or runway closed due to weather conditions

Airport and Government Authorities		
85	MANDATORY SECURITY	Passengers, baggage, crew, etc.
86	IMMIGRATION, CUSTOMS, HEALTH	Passengers, crew
87	AIRPORT FACILITIES	Parking stands, ramp congestion, lighting, buildings, gate limitations etc.
88	RESTRICTIONS AT DESTINATION AIRPORT	Airport and/or runway closed due to obstruction industrial action, staff shortage, political unrest, noise abatement, night curfew, special flights
89	RESTRICTIONS AT AIRPORT OF DEPARTURE	Including air traffic services, start-up and push-back, airport and/or runway closed due to obstruction or weather (restriction due to weather in case of ATFM only) industrial action, staff shortage, political unrest, noise abatement, night curfew, special flights

Reactionary		
91	LOAD CONNECTION	Awaiting load from another flight
92	THROUGH CHECK-IN ERROR	Passenger or baggage check-in error at originating station
93	AIRCRAFT ROTATION	Late arrival of aircraft from another flight or previous sector
94	CABIN CREW ROTATION	Awaiting cabin crew from another flight
95	CREW ROTATION	Awaiting flight deck, or entire crew, from another flight
96	OPERATIONS CONTROL	Re-routing, diversion, consolidation, aircraft change for reasons other than technical

Miscellaneous		
97	INDUSTRIAL ACTION WITHIN OWN AIRLINE	Industrial action (includes Air Traffic Control Services)
98	INDUSTRIAL ACTION OUTSIDE OWN AIRLINE	Industrial action (except Air Traffic Control Services)

99	MISCELLANEOUS	No suitable code; explain reason(s) in plain text

D.2 TAP Delay Codes and Labels

Table D.2: TAP delay codes and related labels.

code	label
829	LACK OF OR LATE FUEL TRUCK
831	AIRCRAFT DEFECTS AT HOME BASE
832	AIRCRAFT DEFECTS AT OUTSTATIONS
833	LACK OF STAFF
835	X-RAY BAGGAGE SCANNING
836	SUSPEND MISSING PAX BAG SEARCH
837	OPERATIONAL SECURITY INSPECTION
838	ACC OCC-PAX IRREG
841	SLOW BOARD ON PREVIOUS FLT
843	LOADING/UNLOADING DELAYED ON PREV FLT
848	AIRCFT DEF ON PREV FLT
849	DCS DELAYED ON PREV FLT
850	LACK OF FLT CREW ON PREV FLT
851	LATE CREW BOARD ON PREV FLT
852	WEATHER COND ON PREV FLT
853	AIR TRAFFIC SERVICES ON PREV FLT
854	SECURITY DELAY ON PREV FLT
855	AIRPORT FACILITIES ON PREV FLT
856	LOAD CONNECTION PREV FLT
857	FLT/BLOCK TIME OF PREV FLT
858	ROTATION OTHERS
860	PASSENGER
861	WEATHER AT STATION OF DEPARTURE
865	LATE BAGGAGE ACCEPTANCE
868	SEARCHING/OFF LOADING MISSING PAX BAG
870	COMMERCIAL REASONS, PUBLICITY, PAX'S CON
871	AIRPORT SLOT
877	BAGGAGE
884	ULD LACK OF/OR SERVICE ABILITY
885	PGA OPS CONTROL
887	HCC
888	LATE CHECK-IN/ACEPTANCE AFTER DEADLINE
893	LATE CHECK-IN/CONGESTION CHECK AREA
897	CHECK-IN ERROR/PASSENGER AND BAGGAGE
904	SLOW BOARDING,DISCREPANCIES AND PAGING
905	SLOW BOARDING/GATE ERROR, LACK OF STAFF
908	EXCESSIVE HAND LUGGAGE
909	ILLNESS/DEATH OF PAX
911	LATE OR WRONG DELIVERY FROM DEPARTURE HA
912	LATE OR WRONG DELIVERY FROM TRANSFER HAL
914	BAGGAGE PROCESSING, SORTING, ETC.
919	DOCUMENTATION, ERRORS, ETC.

923	LATE ACCEPTANCE
936	AIRCRAFT DOCUMENTATTION LATE/INACCURATE
940	CABIN CLEANING
941	LOADING/UNLOADING
942	LACK OF LOADING STAFF, ERROR
946	LOADING EQUIPMENT
947	SERVICING EQUIPMENT
948	LACK OF OR LATE PAX. STAIRS
949	LACK OF OR LATE PAX. BUS
951	AIRCRAFT CLEANNING
952	FUELLING/DEFUELLING
953	CATERING, LATE DELIVERY OR LOADING DISCR
954	LATE CATERING DELIVERY BY CATERING COMPA
955	DISCREPANCIES
956	AIRCRAFT CHANGE, MEAL PLAN
959	TECHNICAL EQUIPMENT
963	AIRCRAFT DEFECTS
964	SCHEDULED MAINTENANCE/LATE RELEASE
968	NON SCHEDULED MAINTENANCE
975	AIRCRAFT CHANGE FOR TECHNICAL REASONS
976	AIRCRAFT CHANGE DUE AIRCRAFT DEFECT
980	DAMAGE DURING GROUND OPERATION
984	DEPARTURE CONTROL SYSTEM
985	DCS ERROR
987	FLIGHT PLANS
988	OTHER SYSTEMS
990	DATA LINE INTERRUPTED
991	OPERATIONAL REQUIREMENTS
992	LATE CREW BOARD/DEPARTURE PROCEDURES
993	ENTIRE CREW LATE BOARDING
998	FLIGHT DECK CREW SHORTAGE
999	FLIGHT DECK CREW SPECIAL REQUEST
1000	CABIN CREW BOARDING
1001	CABIN CREW SHORTAGE
1002	CABIN CREW ERROR OR SPECIAL REQUEST
1003	WRONG HEAD CHECK
1004	RE-ORDERS
1006	WEATHER AT STATION OF DESTINATION
1008	ATFM DUE TO ATC EN-ROUTE DEMAND/CAPACITY
1009	DE-ICING OF AIRCRAFT
1011	GROUND HANDLING IMPAIRED
1013	ATFM DUE TO RESTRICTION AT DEST APT
1014	ATFM DUE TO WEATHER AT DESTINATION
1015	MANDATORY SEGURITY
1016	IMMIGRATION, CUSTOMS
1017	AIRPORT FACILITIES
1018	RESTRICTIONS AT AIRPORT OF DEPARTURE
1019	LOAD CONNECTION (PAX/CARGO/MAIL)
1021	AIRCRAFT ROTATION
1022	CABIN CREW ROTATION
1023	CREW ROTATION

1024	ENTIRE CREW TOO LATE DUE TO ROTATION
1025	FLIGHT CREW TOO LATE DUE TO ROTATION
1026	OPERATIONS CONTROL
1028	INDUSTRIAL ACTION OUTSIDE OWN AIRLINE
1029	NOT ELSEWHERE SPECIFIED
1031	DPG (Planning and Management)
1032	SCHEDULED GROUND TIME LESS THAN DECLARED
1035	LATE AIRCRAFT DOCUMENT BRIEFCASE

D.3 IATA and TAP Delay Code Classification

Table D.3 Manual IATA delay code classification.

code	flight problem	crew problem	code	flight problem	crew problem	code	flight problem	crew problem	code	flight problem	crew problem
6	AIRP	–	**9**	OTH	–	**11**	COMM	–	**12**	COMM	–
13	COMM	–	**14**	HAND	–	**15**	COMM	–	**16**	HAND	–
17	HAND	–	**18**	SEC	–	**21**	HAND	–	**22**	HAND	–
23	HAND	–	**24**	HAND	–	**25**	HAND	–	**27**	HAND	–
28	HAND	–	**29**	HAND	–	**31**	HAND	–	**32**	AIRP	–
33	AIRP	–	**34**	AIRP	–	**35**	AIRP	–	**36**	AIRP	–
37	AIRP	–	**38**	AIRP	–	**39**	AIRP	–	**41**	MAINT	–
42	MAINT	–	**43**	MAINT	–	**44**	MAINT	–	**45**	MAINT	–
46	OTH	–	**47**	MAINT	–	**51**	MAINT	–	**52**	MAINT	–
55	ATC	–	**56**	HAND	–	**57**	ATC	–	**61**	ROT	–
62	ATC	–	**63**	CREW	INDUTY	**64**	CREW	SIGN	**65**	CREW	OTH
66	CREW	INDUTY	**67**	CREW	SIGN	**68**	CREW	OTH	**69**	CREW	OTH
71	ATC	–	**72**	ATC	–	**73**	ATC	–	**75**	AIRP	–
76	AIRP	–	**77**	METEO	–	**81**	ATC	–	**82**	ATC	–
83	ATC	–	**84**	METEO	–	**85**	SEC	–	**86**	SEC	–
87	AIRP	–	**88**	ATC	–	**89**	ATC	–	**91**	ROT	–
92	SEC	–	**93**	ROT	–	**94**	CREW	ROT	**95**	CREW	ROT
96	ATC	–	**97**	OTH	–	**98**	OTH	–	**99**	OTH	–

Table D.4 Manual TAP delay code classification.

code	flight problem	crew problem	code	flight problem	crew problem	code	flight problem	crew problem	code	flight problem	crew problem
829	AIRP	–	**831**	MAINT	–	**832**	MAINT	–	**833**	AIRP	–
835	SEC	–	**836**	SEC	–	**837**	SEC	–	**838**	COMM	–
841	HAND	–	**843**	HAND	–	**848**	MAINT	–	**849**	MAINT	–
850	CREW	ROT	**851**	CREW	ROT	**852**	METEO	–	**853**	ATC	–
854	SEC	–	**855**	AIRP	–	**856**	HAND	–	**857**	ROT	–
858	ROT	–	**860**	COMM	–	**861**	METEO	–	**865**	COMM	–
868	HAND	–	**870**	HAND	–	**871**	AIRP	–	**877**	HAND	–
884	AIRP	–	**885**	AIRP	–	**887**	AIRP	–	**888**	COMM	–
893	COMM	–	**897**	HAND	–	**904**	COMM	–	**905**	HAND	–
908	SEC	–	**909**	COMM	–	**911**	HAND	–	**912**	HAND	–
914	HAND	–	**919**	HAND	–	**923**	HAND	–	**936**	HAND	–
940	AIRP	–	**941**	AIRP	–	**942**	AIRP	–	**946**	AIRP	–
947	AIRP	–	**948**	AIRP	–	**949**	AIRP	–	**951**	AIRP	–
952	AIRP	–	**953**	AIRP	–	**954**	AIRP	–	**955**	OTH	–
956	OTH	–	**959**	MAINT	–	**963**	MAINT	–	**964**	MAINT	–
968	MAINT	–	**975**	MAINT	–	**976**	MAINT	–	**980**	MAINT	–
984	ATC	–	**985**	ATC	–	**987**	ATC	–	**988**	OTH	–
990	ATC	–	**991**	OTH	–	**992**	CREW	INDUTY	**993**	CREW	INDUTY
998	CREW	SIGN	**999**	CREW	OTH	**1000**	CREW	OTH	**1001**	CREW	SIGN
1002	CREW	OTH	**1003**	COMM	–	**1004**	ROT	–	**1006**	ATC	–
1008	ATC	–	**1009**	AIRP	–	**1011**	METEO	–	**1013**	ATC	–
1014	METEO	–	**1015**	SEC	–	**1016**	SEC	–	**1017**	AIRP	–
1018	ATC	–	**1019**	ROT	–	**1021**	ROT	–	**1022**	CREW	ROT
1023	CREW	ROT	**1024**	CREW	ROT	**1025**	CREW	ROT	**1026**	ATC	–
1028	OTH	–	**1029**	OTH	–	**1031**	CREW	RULES	**1032**	OTH	–
1035	OTH	–									

References

Abdelghany, Ahmed F., Ekollu, Goutham, Narasimhan, Ram, & Abdelghany, Khaled F. 2004. A Proactive Crew Recovery Decision Support Tool for Commercial Airlines During Irregular Operations. *Annals OR*, **127**(1-4), 309–331.

Abdelghany, Khaled F., Abdelghany, Ahmed F., & Ekollu, Goutham. 2008. An integrated decision support tool for airlines schedule recovery during irregular operations. *European Journal of Operational Research*, **185**(2), 825 – 848.

Adacher, Ludovica, Boccadoro, Mauro, Martinelli, Francesco, & Valigi, Paolo. 2008. Cooperative and competitive negotiation in a Supply Chain model. *Pages 3731–3736 of: CDC'08*.

AEA. 2010. *Delivering a Bright Future for European Aviation and Passengers - 5 Year Strategic Plan 2010-2014*. Tech. rept. Association of European Airlines, Brussels, Belgium.

Aknine, Samir, Pinson, Suzanne, & Shakun, Melvin F. 2004. An Extended Multi-Agent Negotiation Protocol. *Autonomous Agents and Multi-Agent Systems*, **8**(1), 5–45.

Andersson, T., & Varbrand, P. 2004. The flight perturbation problem. *Transportation Planning and Technology*, **27**(2), 91–117.

Andersson, Tobias. 2001. *The flight perturbation problem - operational aircraft rescheduling*. PhD Thesis, Linkoping University, Sweden.

Andersson, Tobias. 2006. Solving the flight perturbation problem with meta heuristics. *Journal of Heuristics*, **12**(1-2), 37–53.

Arguello, M.F., Bard, J.F., & Yu, G. 1997. A Grasp for Aircraft Routing in Response to Groundings and Delays. *Journal of Combinatorial Optimization*, **1**(3), 211–228(18).

Bard, J.F., Yu, G., & Arguello, M.F. 2001. Optimizing Aircraft Routings in response to Groundings and Delays. *IIE Transactions*, **33**(10), 931–947(17).

Barnhart, Cynthia, Belobaba, Peter, & Odoni, Amedeo R. 2003. Applications of Operations Research in the Air Transport Industry. *TRANSPORTATION SCIENCE*, **37**(4), 368–391.

Bartolini, Claudio, & Preist, Chris. 2001. *A Framework for Automated Negotiation*. Tech. rept. HPL-2001-90. HP Labs.

Bauer, Bernhard, & Odell, James. 2005. UML 2.0 and agents: How to build agent-based systems with the new UML standard. *Journal of Enginnering Applications of Artificial Intelligence*, **18**(2), pp. 141–157.

Bellifemine, F., Caire, G., Trucco, T., & Rimassa, G. 2004. *JADE Programmers Guide, JADE 3.3*. TILab S.p.A.

Bergenti, Federico, Gleizes, Marie-Pierre, & Zambonelli, Franco (eds). 2004. *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*. Kluwer Academic Publishers.

Bisaillon, Serge, Cordeau, Jean-François, Laporte, Gilbert, & Pasin, Federico. 2010. A large neighbourhood search heuristic for the aircraft and passenger recovery problem. *4OR: A Quarterly Journal of Operations Research*, 1–19. 10.1007/s10288-010-0145-5.

Brams, Steven J. 2003. *Negotiation Games: Applying Game Theory to Bargaining and Arbitration*. Routledge Publishers.

Bratu, Stephane, & Barnhart, Cynthia. 2006. Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling*, **9**(3), 279–298.

Bresciani, Paolo, Perini, Anna, Giorgini, Paolo, Giunchiglia, Fausto, & Mylopoulos, John. 2004. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, **8**(3), 203–236.

Caire, G., Coulier, W., Garijo, F., Gomez, J., Pavon, J., Massonet, P., & Leal, F. 2001. Agent Oriented Analysis Using MESSAGE/UML. *In: Proceedings of Agent Oriented Software Engineering (AOSE 2001)*. Springer.

Cao, Jia-Ming, & Kanafani, Adib. 1997a. Real-time decision support for integration of airline flight cancellations and delays part I: mathematical formulation. *Transportation Planning and Technology*, **20**(3), 183.199.

Cao, Jia-Ming, & Kanafani, Adib. 1997b. Real-time decision support for integration of airline flight cancellations and delays. Part II: Algorithm and computational experiments. *Transportation Planning and Technology*, **20**(3), 201–217.

Carlos, Tom. 2012 (December). *Requirements Traceability atrix - RTM (web site)*.

Castro, Antonio J. M. 2007. *Designing a Multi-Agent System for Monitoring and Operations Recovery for an Airline Operations Control Centre*. MsC Thesis, University of Porto, Faculty of Engineering, Porto, Portugal.

Castro, Antonio J. M. 2008 (October). *Centros de Controlo Operacional: Organizacao e Ferramentas*. ISEC - Instituto Superior de Educacao e Ciencias. Monograph for Post-graduation in Air Transport Operations.

Castro, Antonio J. M., & Oliveira, Eugenio. 2007. Using Specialized Agents in a Distributed MAS to Solve Airline Operations Problems: A Case Study. *Pages 473–476 of: IAT '07: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*. Washington, DC, USA: IEEE Computer Society.

Castro, Antonio J. M., & Oliveira, Eugenio. 2008. The rationale behind the development of an airline operations control centre using Gaia based methodology. *International Journal of Agent-Oriented Software Engineering*, **2**(3), 350–377.

Castro, Antonio J. M., & Oliveira, Eugenio. 2009. Quantifying Quality Operational Costs in a Multi-Agent System for Airline Operations Recovery. *International Review on Computers and Software (IRECOS)*, **4**(4), 504–516.

Castro, Antonio J. M., & Oliveira, Eugenio. 2010. *Web Intelligence and Intelligent Agents*. INTECH. Chap. Disruption Management in Airline Operations Control An Intelligent Agent-Based Approach, pages 107–132.

Castro, Antonio J. M., & Oliveira, Eugenio. 2011. A New Concept for Disruption Management in Airline Operations Control. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, **3**(3), 269–290.

Castro, Antonio J. M., Rocha, Ana Paula, & Oliveira, Eugenio. 2012. Towards an Autonomous and Intelligent Airline Operations Control. *Pages 1429–1434 of: Proceedings of the 2012 15th IEEE Conference on Intelligent Transportation Systems (ITSC 2012)*. Anchorage, Alaska, USA: IEEE Publisher.

Cernuzzi, L., & Zambonelli, F. 2004. Experiencing AUML in the GAIA Methodology. *Pages 283–288 of: Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS)*.

Chen, Xindu, Chen, Xin, & Zhang, Xinhui. 2010. Crew scheduling models in airline disruption management. *Pages 1032–1037 of: 2010 IEEE 17Th International Conference on Industrial Engineering and Engineering Management (IE&EM)*. Conference Publications.

Clarke, Michael Dudley Delano. 1997 (October). *The airline schedule recovery problem*. Working Paper.

Clarke, Michael Dudley Delano. 1998. *Development of heuristic procedures for flight rescheduling in the aftermath of irregular airline operations*. PhD Thesis, Massachusetts Institute of Technology. Dept. of Aeronautics and Astronautics, USA.

Clausen, J., Larsen, A., & Larsen, J. 2005. *Disruption Management in the Airline Industry - Concepts, Models and Methods*. Technical Report. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby.

Clausen, Jens, Larsen, Allan, Larsen, Jesper, & Rezanova, Natalia J. 2010. Disruption management in the airline industry-Concepts, models and methods. *Computers and Operations Research*, **37**(5), 809–821.

CODA. 2012. *CODA Digest Delays to Air Transport in Europe Annual 2011*. Tech. rept. EUROCONTROL, Brussels, Belgium.

Coleman, James Samuel. 1973. *The Mathematics of Collective Action*. Transaction Publishers.

Colorni, A., Dorigo, M., & Maniezzo, V. 1991. Distributed Optimization by Ant Colonies. *Pages 134–142 of: Actes de la Prémiere Conférence Europeenne sur la Vie Artificielle*. Paris, France: Elsevier Publishing.

Cook, Andrew, & Tanner, Graham. 2011 (March). *European Airline Delay Cost Reference Values*. Technical Report. Department of Transport Studies, University of Westminster, London.

Cook, Andrew, Tanner, Graham, & Anderson, Stephen. 2004 (February). *Evaluating the True Cost to Airlines of One Minute of Airborne or Ground Delay*. Technical Report. Transport Studies Group, University of Westminster, London.

Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., & Stein, Clifford. 2001. *Introduction to Algorithms*. 2nd edn. MIT Press and McGraw-Hill.

Cossentino, M., & Potts, C. 2002. A CASE tool supported methodology for the design of multi-agent systems. *In: Proceedings of the 2002 International Conference on Software Engineering Research and Practice (SERP02)*.

Cossentino, M., Burrafato, P., Lombardo, S., & Sabatucci, L. 2003. *Agent Technologies, Infrastructures, Tools and Applications for E-Services*. LNAI 2592. Berlin: Springer-Verlag. Chap. Introducing Pattern Reuse in the Design of Multi-Agent Systems, pages 107–120.

Crawford, E., & Veloso, M. 2008 (December). Negotiation in Semi-cooperative Agreement Problems. *Pages 252 –258 of: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT '08*, vol. 2.

Dang, Jiangbo, & Huhns, Michael N. 2005. An extended protocol for multiple-issue concurrent negotiation. *Pages 65–70 of: AAAI'05: Proceedings of the 20th national conference on Artificial intelligence*. AAAI Press.

D'Ariano, Andrea, & Hermelrijk, Robin. 2006. Designing a Multi-Agent System for Cooperative Train Dispatching. *Information Control Problems in Manufacturing*, **12**.

Dijkstra, Edsger W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, 269–271.

Dimopoulos, Yannis, & Moraitis, Pavlos. 2010. *Negotiation and Argumentation in MAS*. Bentham Science Publishers Ltd. Chap. Advances in Argumentation-based Negotiation, pages 1–44.

Dorigo, Marco. 1992. *Optimization, Learning and Natural Algorithms*. Phd Thesis, Politecnico di Milano, Italie.

Durfee, E. H., Lesser, V. R., & Corkill, D. D. 1989. Trends in Cooperative Distributed Problem Solving. *IEEE Transactions on Knowledge and Data Engineering*, **1**(1), 63–83.

Eggenberg, Niklaus, Salani, Matteo, & Bierlaire, Michel. 2007. *Column Generation Methods for Disrupted Airline Schedules*. Fifth Joint Operations Research Days, ETHZ, Zurich, August 28, 2007.

Eggenberg, Niklaus, Salani, Matteo, & Bierlaire, Michel. 2010. Constraint-specific recovery network for solving airline recovery problems. *Computers and Operations Research*, **37**(6), 1014–1026.

Elamy, A. 2005 (August). *Perspectives in Agents-Based Technology*. AgentLink News 18.

Fabio Bellifemine, Giovanni Caire, & Greenwood, Dominic. 2007. *Developing Multi-Agent Systems with JADE*. Wiley Series in Agent Technology. John Wiley & Sons. Chap. Chapter 9 - Deploying a Fault-Tolerant JADE Platform, pages 173–179.

Faratin, P., Sierra, C., & Jennings, N.R. 2002. Using similarity criteria to make issue trade-offs in automated negotiations. *Artificial Intelligence*, **142**(2), 205 – 237. International Conference on MultiAgent Systems 2000.

Feo, Thomas A., & Resende, Mauricio G. C. 1989. A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem. *Operations Research Letters*, **8**(April), 67–71.

Finin, Tim, Fritzson, Richard, McKay, Don, & McEntire, Robin. 1994. KQML as an agent communication language. *Pages 456–463 of: Proceedings of the third international conference on Information and knowledge management*. CIKM '94. New York, NY, USA: ACM.

FIPA. 2002a (December). *FIPA Communicative Act Library Specification*.

FIPA. 2002b. *FIPA Contract Net Interaction Protocol Specification*.

FIPA. 2002c. *FIPA Request Interaction Protocol Specification*.

Fox, Mark S. 1981. An Organizational View of Distributed Systems. *IEEE Transactions on Systems, Man and Cybernetics*, **11**(1), 70 –80.

Gao, Chunhua. 2007 (August). *Airline Integrated Planning and Operations*. Ph.D. thesis, Georgia Institute of Technology.

García-Ojeda, J., Arenas, A., & Pérez-Alcázar, J. 2005. Paving the Way for Implementing Multi-Agent Systems: Refining GAIA with AUML. *Pages 179–189 of: Proceedings of the 6th International Workshop on Agent-Oriented Software Engineering (AOSE 2005)*. LNCS 3950. Berlin: Springer-Verlag.

Glover, Fred. 1989. Tabu Search Part I. *ORSA Journal on Computing*, **1**(3), 190–206.

Glover, Fred. 1990. Tabu Search Part II. *ORSA Journal on Computing*, **2**(1), 4–32.

Goldman, Claudia V., & Rosenschein, Jeffrey S. 1995. Mutually Supervised Learning in Multi-agent Systems. *Pages 85–96 of: Proceedings of the Workshop on Adaption and Learning in Multiagent Systems (IJCAI95)*. Springer-Verlag.

Goradia, Hrishikesh J. 2007. *Automated Negotiations Among Autonomous Agents in Negotiation Networks*. Ph.D. thesis, Department of Computer Science and Engineering, College of Engineering and Computing, University of South Carolina.

Grosche, Tobias. 2009. *Computational Intelligence in Integrated Airline scheduling*. Germany: Springer-Verlag Berlin Heidelberg.

Grossman, Dan. 2013 (January). *DELAG: The Worlds First Airline (web site)*.

Group, Transport Studies. 2008 (June, 2). *Dynamic Cost Indexing: Aircraft Maintenance - Marginal Delay Costs*. Technical Discussion Document 9.0 C06/12400BE. University of Westminster, London, UK.

Guo, Yufeng. 2004. A Decision Support Framework for the Airline Crew Schedule Disruption Management with Strategy Mapping. *Pages 158–165 of:* Fleuren, Hein A., den Hertog, Dick, & Kort, Peter M. (eds), *Operations Research, Proceedings 2004, Selected Papers of the Annual International Conference of the German Operations Research Society (GOR)*.

Hemaissia-Jeannin, Miniar, Seghrouchni, Amal El Fallah, & Labreuche, Christophe. 2008. A new multilateral multi-issue negotiation protocol and its application to a crisis management problem. *Multiagent and Grid Systems*, **4**(1), 103–123.

Henderson-Sellers, Brian, & Giorgini, Paolo (eds). 2005. *Agent-Oriented Methodologies*. Idea Group Publishing.

Hendler, J., & McGuiness, D. L. 2000. The DARPA Agent Markup Language. *IEEE Intelligent Systems*, **15**(6), 67–73.

Hoen, Pieter, Tuyls, Karl, Panait, Liviu, Luke, Sean, & La Poutré, J. 2006. An Overview of Cooperative and Competitive Multiagent Learning. *Pages 1–46 of:* Tuyls, Karl, Hoen, Pieter, Verbeeck, Katja, & Sen, Sandip (eds), *Learning and Adaption in Multi-Agent Systems*. Lecture Notes in Computer Science, vol. 3898. Springer Berlin / Heidelberg. 10.1007/11691839-1.

Homans, George Gaspar. 1973. *Social Behaviour: its elementary forms*. Taylor and Francis Publisher.

IATA. 2001. *Airport and Air Navigation Charges Manual*. International Air Transport Association, Montreal, Geneva, Switzerland.

Irrang, M. E. 1996. *The Handbook of Airline Economics*. Air Transport Association of America. Chap. Airline Irregular Operations, pages 349–365.

Jafari, Niloofar, & Zegordi, Seyed Hessameddin. 2010. The Airline Perturbation Problem: Considering Disrupted Passengers. *Transportation Planning and Technology*, **33**(2), 203–220.

Jain, Vipul, & Deshmukh, S.G. 2009. Dynamic supply chain modeling using a new fuzzy hybrid negotiation mechanism. *International Journal of Production Economics*, **122**(1), 319 – 328. Transport Logistics and Physical DistributionInterlocking of Information Systems for International Supply and Demand Chains Management ICPR19.

Jarrah, Ahmad I. Z., Yu, Gang, Krishnamurthy, Nirup, & Rakshit, Ananda. 1993. A Decision Support Framework for Airline Flight Cancellations and Delays. *TRANSPORTATION SCIENCE*, **27**(3), 266–280.

Jennings, Nicholas R., Faratin, P., Lomuscio, A. R., Parsons, S., Wooldridge, M., & Sierra, C. 2001. Automated Negotiation: Prospects, Methods and Challenges. *Group Decision and Negotiation, Kluwer Academic Publishers*, **10**, 199–215.

Ji, Guojun, & Zhu, Caihong. 2008. Study on supply chain disruption risk management strategies and model. *Pages 1–6 of: 2008 International Conference on Service Systems and Service Management*.

Joana Urbano, Ana Paula Rocha, Eugénio Oliveira. 2012. Trust Evaluation for Reliable Electronic Transactions Between Business Partners. *In:* Fischer, Klaus, Müller, Jörg P., & Levy, Renato (eds), *Agent-Based Technologies and Applications for Enterprise Interoperability*. Lecture Notes in Business Information Processing, vol. 98. Springer. International Workshops ATOP 2009, Budapest, Hungary, May 12, 2009, and ATOP 2010, Toronto, Canada, May 10, 2010, Revised Selected Papers.

Johnson, V., Lettovsky, L., Nemhauser, G.L., Pandit, R., & Querido, S. 1994. *Final report to Northwest Airlines on the crew recovery problem*. Technical Report. The Logistic Institute, Georgia Institute of Technology, Atlanta, USA.

Jonker, Geert, Meyer, John-Jules Ch., & Dignum, Frank. 2005. Towards a Market Mechanism for Airport Traffic Control. *Pages 500–511 of: EPIA*.

Katsuhide Fujita, Takayuki Ito, & Klein, Mark. 2010. *Web Intelligence and Intelligent Agents*. InTech. Chap. Representative-based Protocol for Multiple Interdependent Issue Negotiation Problems, pages 347–364.

Kazakov, Dimitar, & Kudenko, Daniel. 2001. Machine Learning and Inductive Logic Programming for Multi-Agent Systems. *Pages 246–270 of: Multi-Agent Systems and Applications*. Springer.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. 1983. Optimization by Simulated Annealing. *Science*, **220**(4598), 671–680.

Kohl, N., Larsen, A., Larsen, J., Ross, A., & Tiourline, S. 2004. *Airline Disruption Management: Perspectives, Experiences and Outlook*. Technical Report CRTR-0407. Carmen Research.

Kohl, Niklas, & Karisch, Stefan E. 2004. Airline Crew Rostering: Problem Types, Modeling, and Optimization. *Annals OR*, **127**(1-4), 223–257.

Leitão, Paulo. 2011. A holonic disturbance management architecture for flexible manufacturing systems. *International Journal of Production Research*, **49**(5), 1269–1284.

Leslie Pack Kaelbling, Michael L. Littman, & Moore, Andrew W. 1996. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, **4**, 237–285.

Lettovsky, L. 1997. *Airline Operations Recovery: An Optimization Approach*. PhD Thesis, Georgia Institute of Technology, Atlanta, USA.

Lettovsky, Ladislav, Johnson, Ellis L., & Nemhauser, George L. 2000. Airline Crew Recovery. *TRANSPORTATION SCIENCE*, **34**(4), 337–348.

Lin, Raz, Kraus, Sarit, Baarslag, Tim, Tykhonov, Dmytro, Hindriks, Koen, & Jonker, Catholijn M. 2011. GENIUS: An Integrated Environment for Supporting the Design of Generic Automated Negotiators. *Computational Intelligence*.

Liu, S. F., & Guo, T. B. 1992. *The Grey System Theory and Application*. Kaifeng: Science Press.

Liu, Tung-Kuan, Jeng, Chi-Ruey, Liu, Yu-Ting, & Tzeng, Jia-Ying. 2006 (October). Applications of Multi-objective Evolutionary Algorithm to Airline Disruption Management. *Pages 4130 – 4135 of: IEEE International Conference on Systems, Man and Cybernetics, 2006. SMC '06*, vol. 5.

Liu, Tung-Kuan, Jeng, Chi-Ruey, & Chang, Yu-Hern. 2008. Disruption Management of an Inequality-Based Multi-Fleet Airline Schedule by a Multi-Objective Genetic Algorithm. *Transportation Planning and Technology*, **31**(6), 613–639.

Lomuscio, Alessio R., Wooldridge, Michael, & Jennings, Nicholas R. 2000. A Classification Scheme for Negotiation in Electronic Commerce. *International Journal Group Decision and Negotiation*, **12**(1), 31–56.

Lopes, Fernando, Mamede, Nuno, Novais, A. Q., & Coelho, Helder. 2002. A negotiation model for autonomous computational agents: Formal description and empirical evaluation. *Journal Intelligent Fuzzy Systems*, **12**(3,4), 195–212.

Lopes, Fernando, Novais, A. Q., Mamede, Nuno, & Coelho, Helder. 2005. A negotiation model for autonomous agents: key features and comparison with existing models. *Pages 1211–1212 of: Proceedings of the fourth international joint conference on Autonomousagents and multiagent systems*. AAMAS '05. New York, NY, USA: ACM.

Lopes, O, Mamede, Nuno, Novais, A. Q., & Coelho, Helder. 2004. Negotiation strategies for autonomous computational agents. *Pages 38–42 of: In ECAI-04*. IOS Press.

Lopes Cardoso, Henrique, Urbano, Joana, Rocha, Ana Paula, Castro, António J. M., & Oliveira, Eugénio. 2013. *Agreement Technologies*. Law, Governance and Techonology, vol. 8. Springer. Chap. ANTE: Agreement Negotiation in Normative and Trust-enabled Environments (Chapter 32), pages 549–564.

Løve, M., Sørensen, K. R., Larsen, J., & Clausen, J. 2005. Using Heuristics to Solve the Dedicated Aircraft Recovery Problem. *Central European Journal of Operations Research*, **13**(2), 189–207.

Luck, M., McBurney, P., Shehory, O., & Willmott, S. 2005. *Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing)*. AgentLink.

Luo, Songjun, & Yu, Gang. 1997a. On the Airline Schedule Perturbation Problem Caused by the Ground Delay Program. *TRANSPORTATION SCIENCE*, **31**(4), 298–311.

Luo, Songjun, & Yu, Gang. 1997b. *Operations Research in the Airline Industry*. 1st edn. Springer. Chap. Airline schedule perturbation problem: landing and take-off with nonsplitable resource for the ground delay program, pages 404–432.

Luo, X., Jennings, N. R., Shadbolt, N., Leung, H., & Lee, J.H. 2003. A fuzzy constraint based model for bilateral multi-issue negotiations in semi-competitive environments. *Artificial Intelligence Journal*, **148**(1-2), 53–102.

Machado, Nuno Goncalo Sobral Gomes Amaral. 2010 (July). *Impact of the Organizational Structure on Operations Management: The Airline Operations Case Study*. MsC Thesis, Faculty of Engineering, University of Porto, Porto, Portugal.

Malucelli, Andreia, Castro, Antonio J. M., & Oliveira, Eugenio. 2006. Crew and Aircraft Recovery Through a Multi-Agent Electronic Market. *Pages 51–58 of:* Krishnamurthy, Sandeep, & Isaias, Pedro (eds), *Proceeding of IADIS International Conference on e-Commerce 2006*. Barcelona Spain: IADIS Press.

March, J. G., & Simon, H. A. 1993. *Organizations*. 2nd edition (1st edition 1958) edn. New York, USA.: John Wiley & Sons.

Mathaisel, Dennis F. X. 1996. Decision support for airline system operations control and irregular operations. *Computers and Operations Research*, **23**(11), 1083–1098.

McConnell, Steve. 2004. *Code Complete*. 2nd edn. Microsoft Press.

McGuiness, D. L., & van Harmelen, F. 2003. *Web ontology language (OWL): overview.* Tech. rept. W3C.

Medard, Claude P., & Sawhney, Nidhi. 2007. Airline crew scheduling from planning to operations. *European Journal of Operational Research*, **183**(3), 1013 – 1027.

Moraitis, P., & Spanoudakis, N. 2006. The GAIA2JADE Process for Multi-Agent Systems Development. *Applied Artificial Intelligence*, 251–273.

Moraitis, P., Petraki, E., & Spanoudakis, N. 2003a. *Agent Technologies, Infrastructures, Tools and Applications for E-Services*. LNAI 2592. Berlin: Springer-Verlag. Chap. Engineering JADE agents with the GAIA methodology, pages 77–91.

Moraitis, P., Petraki, E., & Spanoudakis, N. 2003b. Providing Advanced, Personalized Infomobility Services Using Agent Technology. *Pages 35–48 of: Proceedings of the 23rd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (AI 2003)*.

Morgado, Ernesto, & Martins, Joao Pavao. 2012 (June). Autmoated Real-time Dispatching Support. *In: Proceedings of the 2012 Rail Conference*. American Public Transportation Association, Dallas, Texas, USA.

Mu, Q., Fu, Z., Lysgaard, J., & Eglese, R. 2011. Disruption Management of the Vehicle Routing Problem with Vehicle Breakdown. *Journal of the Operational Research Society*, **62**, 742–749.

Nemhauser, George L., Savelsbergh, Martin W. P., & Sigismondi, Gabriele C. 1994. MINTO, a mixed INTeger optimizer. *Operations Research Letters*, **15**(1), 47–58.

Newman, Michael. 2002 (June). *Software Errors Cost U.S. Economy USD59.5 Billion Annually*. Technical Report. National Institute of Standards and Technology.

Nissen, Rüdiger, & Haase, Knut. 2006. Duty-period-based network model for crew rescheduling in European airlines. *J. of Scheduling*, **9**(3), 255–278.

Oliveira, Eugénio. 2010. *MIC - Metodologias de Investigação Científica*. Sebenta da Aula de Metodologias de Investigação Científica. FEUP. UP.

Oliveira, Eugénio. 2012 (October 15-16). A Structured Environment to Facilitate Agreements. *Pages 351–352 of: Proceedings of the First International Conference on Agreement Technologies (AT2012), CEUR Workshop Proceedings*.

Osborne, Martin J., & Rubinstein, Ariel. 1994. *A Course in Game Theory: A modern introduction at the graduate level*. MIT Press.

Ossowski, Sascha (ed). 2013. *Agreement Technologies*. Law, Governance and Technology, vol. 8. Springer Verlag.

Ouelhadj, D. 2003 (August). *A Multi-Agent System for the Integrated Dynamic Scheduling of Steel Production*. PhD Thesis, The University of Nottingham, School of Computer Science and Information Technology, England.

Padgham, L., & Winikoff, M. 2002. Prometheus: a Methodology for Developing Intelligent Agents. *In: Proceedings of the 3rd International Workshop on Agent Oriented Software Engineering (AOSE 2002)*.

Parliament, European. 2004 (February). *Regulation (EC) 261/2004 of the European Parliament and Council of 11 February 2004*.

Passos, L. S., Rossetti, R. J. F., & Gabriel, J. 2011 (October, 5-7). An Agent Methodology for Processes, the Environment and Services. *Pages 2124–2129 of: 14 th International IEEE Conference on Intelligent Transportation Systems (ITSC), 2011*. IEEE.

Paulk, Mark C., Curtis, Bill, Chrissis, Mary Beth, & Weber, Charles V. 1993 (February). *Capability Maturity Model for Software (Version 1.1)*. Tech. rept. CMU/SEI-93-TR-024 ESC-TR-93-177. Carnegie Mellon University.

Pereira, David, Oliveira, Eugenio, Moreira, Nelma, & Sarmento, Luis. 2005. Towards an Architecture for Emotional BDI Agents. *Pages 40–46 of:* Bento, A. Carlos, & Dias, G. (eds), *EPIA 2005. Portuguese conference on Artificial intelligence*. IEEE.

Pereira, David, Oliveira, Eugenio, & Moreira, Nelma. 2008. *Computational Logic in Multi-Agent Systems*. Lecture Notes in Computer Science, vol. 5056/2008. Berlin: Springer Berlin / Heidelberg. Chap. Formal Modelling of Emotions in BDI Agents, pages 62–81.

Petersen, Jon D., Solveling, Gustaf, Johnson, Ellis J., Clarke, Jonh-Paul, & Shebalov, Sergey. 2010 (May). *An Optimization Approach to Airline Integrated Recovery*. Tech. rept. The Airline Group of the International Federation of Operational Research (AGIFORS).

Piques, Paulo. 2006 (November). *A Economia do Transporte Aéreo e a Gestão da sua Capacidade*. Instituto Superior de Educação e Ciências - Post-Graduation in Air Transport Operations (Monograph).

Raffard, Robin L., Waslander, Steven L., Bayen, Alexandre M., & Tomlin, Claire J. 2005 (August). A Cooperative Distributed Approach to Multi-Agent Eulerian Network Control: Application to Air Traffic Management. *In: AIAA Conference on Guidance, Navigation and Control*.

Rahwan, Iyad, Ramchurn, Sarvapalid D., Jennings, Nicholas R., McBurney, Peter, Parsons, Simon, & Sonenberg, Liz. 2004. Argumentation-based Negotiation, Cambridge University Press. *The Knowledge Engineering Review*, **18**(4), 343–375.

Rakshit, Ananda, Krishnamurthy, Nirup, & Yu, Gang. 1996. System Operations Advisor: A Real-Time Decision Support System for Managing Airline Operations at United Airlines. *INTERFACES*, **26**(2), 50–58.

Rocha, Ana Paula, & Oliveira, Eugenio. 1999. An Electronic Market Architecture for the Formation of Virtual Enterprises. *Pages 421–432 of: PRO-VE '99: Proceedings of the IFIP TC5 WG5.3 / PRODNET Working Conference on Infrastructures for Virtual Enterprises*. Deventer, The Netherlands, The Netherlands: Kluwer, B.V.

Rocha, Ana Paula, & Oliveira, Eugenio. 2001. Electronic Institutions as a Framework for Agents' Negotiation and Mutual Commitment. *Pages 232–245 of: EPIA '01: Proceedings of the 10 th Portuguese Conference on Artificial Intelligence on Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving*. London, UK: Springer-Verlag.

Rodrigo Acuna-Agost, Dominique Feillet, Philippe Michelon, & Gueye, Serigne. 2009 (May). *Rescheduling Flights, Aircraft, and Passengers Simultaneously under Disrupted Operations - A Mathematical Programming Approach based on Statistical Analysis*. Tech. rept. The Airline Group of the International Federation of Operational Research Societies (AGIFORS).

Rosenberger, Jay M., Johnson, Ellis L., & Nemhauser, George L. 2003. Rerouting Aircraft for Airline Recovery. *TRANSPORTATION SCIENCE*, **37**(4), 408–421.

Rosenberger, J.M., Schaefer, A.J., Goldsman, D., Johnson, E.L., Kleywegt, A.J., & Nemhauser, G.L. 2000. SimAir: a stochastic model of airline operations. vol. 2.

Rummery, G. A., & Niranjan, M. 1994 (September). *On-line Q-learning using connectionist systems*. CUED/F-INFENG/TR 166. Cambridge University Engineering Department.

Russell, Stuart, & Norvig, Peter. 2003. *Artificial Intelligence: A Modern Approach*. 2nd edn. Prentice Hall. Chap. Chapter 4: Informed Search and Exploration, pages 94–136.

Russell, Stuart, & Norvig, Peter. 2010. *Artificial Intelligence: A Modern Approach*. 3rd edn. Prentice Hall.

Sandholm, T. 1993. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. *11th National Conference on Artificial Intelligence*, January, 256–262.

Sandholm, Thomas W. 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, Massachusetts, London, England: MIT Press. Chap. Distributed Rational Decision Making, pages 201–258.

Searle, John. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press.

Sharp, Tim. 2013 (January). *Worlds First Commercial Airline - The Greatest Moments in Flight (web site)*.

Silva, Daniel Castro, Braga, Rodrigo A. M., Reis, Luis Paulo, & Oliveira, Eugenio. 2012. Designing a meta-model for a generic robotic agent system using Gaia methodology. *Journal of Information Sciences*, **194**(July), 190–210.

Simon, Herbert A. 1955. A Behavioral Model of Rational Choice. *The Quartely Journal of Economics*, **69**(1), 99–118.

Sita. 2010. *Five Steps of Air Travel that Smartphones will Change by 2020*. Tech. rept. SITA.

Smith, R.G. 1980. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, **C-29**(12), 1104–1113.

Smithsonian National Air and Space Museum. 2013 (January). *The World's First Scheduled Airline (web site)*.

Stojkovic, Mirela, & Soumis, Francois. 2001. An Optimization Model for the Simultaneous Operational Flight and Pilot Scheduling Problem. *MANAGEMENT SCIENCE*, **47**(9), 1290–1305.

Stojkovic, Mirela, & Soumis, François. 2005. The operational flight and multi-crew scheduling problem. *Yugoslav Journal of Operations Research*, **15**(1), 25–48.

Stojkovic, Mirela, Soumis, Francois, & Desrosiers, Jacques. 1998. The Operational Airline Crew Scheduling Problem. *Transportation Science*, **32**(3), 232–245.

Stone, Peter, & Veloso, Manuela. 2000. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots*, **8**(3), 345–383.

Sutton, Richard S., & Barto, Andrew G. 1998. *Reinforcement Learning: An Introduction*. The MIT Press.

Talluri, Kalyan T. 1996. Swapping Applications in a Daily Airline Fleet Assignment. *TRANSPORTATION SCIENCE*, **30**(3), 237–248.

TAP. 2011 (July). *TAP Portugal Relatório Anual 2010*.

Teodorovic, Dusan, & Guberinic, Slobodan. 1984. Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operational Research*, **15**(2), 178 – 182.

Teodorovic, Dusan, & Stojkovic, Goran. 1990. Model for operational daily airline scheduling. *Transportation Planning and Technology*, **14**(4), 273–285.

Teodorovic, Dusan, & Stojkovic, Goran. 1995. Model to Reduce Airline Schedule Disturbances. *Journal of Transportation Engineering*, **121**(4), 324–331.

Thengvall, Benjamin G., Yu, Gang, & Bard, Jonathan F. 2001. Multiple fleet aircraft schedule recovery following hub closures. *Transportation Research Part A: Policy and Practice*, **35**(4), 289–308.

Thengvall, Benjamin G., Bard, Jonathan F., & Yu, Gang. 2003. A Bundle Algorithm Approach for the Aircraft Schedule Recovery Problem During Hub Closures. *Transportation Science*, **37**(4), 392–407.

Thengvalla, Benjamin G., Barda, Jonathan F., & Yu, Gang. 2000. Balancing user preferences for aircraft schedule recovery during irregular operations. *IIE Transactions*, **32**(3), 181–193.

Traum, David R. 1999. Speech Acts for Dialogue Agents. *Pages 169–201 of: Foundations of Rational Agency*. Kluwer.

Tumer, Kagan, & Agogino, Adrian K. 2007. Distributed agent-based air traffic flow management. *Page 255 of: AAMAS Proceedings of the Joint Conference on Autonomous Agents and Multiagent Systems*.

Tyler, Tony. 2012 (June). *2012 Annual Review*. Tech. rept. IATA.

Vokřínek, Jiří, Bíba, Jiří, Hodík, Jiří, Vybíhal, Jaromír, & Pěchouček, Michal. 2007. Competitive Contract Net Protocol. *Pages 656–668 of: Proceedings of the 33rd conference on Current Trends in Theory and Practice of Computer Science*. SOFSEM '07. Berlin, Heidelberg: Springer-Verlag.

Watkins, C. J. C. H., & Dayan, Peter. 1992. Q-Learning. *Machine Learning*, **8**, 279–292.

Wei, G., Yu, G., & Song, M. 1997. Optimization Model and Algorithm for Crew Management During Airline Irregular Operations. *Journal of Combinatorial Optimization*, **1**(17), 305–321.

Weiss, Gerhard (ed). 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press.

Williamson, Oliver E. 1983. *Markets and Hierarchies: Analysis and Antitrust Implications*. Free Press.

Wolfe, Shawn R., Enomoto, Francis Y., Jarvis, Peter A., & Sierhuis, Maarten. 2007. Comparing Route Selection Strategies in Collaborative Traffic Flow Management. *Pages 59–62 of: IAT '07: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*. Washington, DC, USA: IEEE Computer Society.

Wollkind, Steven, Valasek, John, & Ioerger, Thomas R. 2004. Automated conflict resolution for air traffic management using cooperative multiagent negotiation. *Pages 2004–4992 of: AIAA Guidance, Navigation, and Control Conference*.

Wood, M., & DeLoach, S. 2001. An Overview of the Multi-Agent Systems Engineering Methodology. *Pages 207–222 of: Proceedings of the Agent-Oriented Software Engineering First International Workshop (AOSE 2000)*. LNCS 1957. Limerick, Ireland: Springer-Verlag.

Wooldridge, M. 2009a. *An Introduction to Multiagent Systems*. 2nd edn. West Sussex, England:: John Wiley & Sons, Ltd. Chap. When is an Agent-Based Solution Appropriate?, pages 183–184.

Wooldridge, M, & Jennings, NR. 1999. The cooperative problem-solving process. *Journal of Logic and Computation*, **9**(4), 563–592.

Wooldridge, Michael. 2009b. *An Introduction to MultiAgent Systems*. 2nd edn. John Wiley & Sons.

Wooldridge, Michael. 2009c. *An Introduction to MultiAgent Systems*. 2nd edn. John Wiley & Sons, Ltd. Chap. Intelligent Agents, pages 21–47.

Wooldridge, Michael. 2009d. *An Introduction to MultiAgent Systems*. 2nd edn. John Wiley & Sons, Ltd. Chap. Working Together, pages 151–181.

Wu, Congcong, & Le, Meilong. 2012. A New Approach to Solve Aircraft Recovery Problem. *Pages 148–154 of: In Proceedings of the Second International Conference on Advanced Communications and Computation (INFOCOMP 2012*. IARIA.

Yan, Shangyao, & Dah-Hwei, Yang. 1996. A decision support framework for handling schedule perturbation. *Transportation Research Part B: Methodological*, **30**(6), 405–419.

Yan, Shangyao, & ping Tu, Yu. 1997. Multifleet routing and multistop flight scheduling for schedule perturbation. *European Journal of Operational Research*, **103**(1), 155 – 169.

Yang, M. 2007 (December). *Using Advanced Tabu Search Techniques to Solve Airline Disruption Management Problems*. PhD Thesis, The University of Texas at Austin.

Yu, Gang, & Qi, Xiangtong. 2004. *Disruption Management: Framework, Models and Applications*. World Scientific Publishing Co. Pte. Ltd.

Yu, Gang, Arguello, Michael, Song, Gao, McCowan, Sandra M., & White, Anna. 2003. A New Era for Crew Recovery at Continental Airlines. *INTERFACES*, **33**(1), 5–22.

Zambonelli, Franco, Jennings, Nicholas R., & Wooldridge, Michael. 2003. Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology*, **12**(3), 317–370.

Zegordi, Seyed Hessameddin, & Jafari, Niloofar. 2010. Solving the Airline Recovery Problem By Using Ant Colony Optimization. *International Journal of Industiral Engineering & Producion Research*, **21**(3), 121–128.

Zeng, Dajun, & Sycara, Katia. 1998. Bayesian learning in negotiation. *International Journal of Human-Computer Studies*, **48**(1), 125–141.

Zhang, Xiaoqin, & Lesser, Victor. 2012. Meta-level Coordination for Solving Distributed Negotiation Chains in Semi-cooperative Multi-agent Systems. *Group Decision and Negotiation*, 1–33. 10.1007/s10726-012-9287-5.

Zhang, Yu, & Hansen, Mark. 2008. Real-Time Intermodal Substitution: Strategy for Airline Recovery from Schedule Perturbation and for Mitigation of Airport Congestion. *Transportation Research Record: Journal of the Transportation Research Board*, **2052**, 90–99.

Zhao, Xiuli, & Guo, Yanchi. 2012. Study on GRAPS-ACO Algorithm for Irregular Flight Rescheduling. *Pages 266–269 of: 2012 International Conference on Computer Acience and Service Systems (CSSS)*. Nanjing, China: IEEE.

Zhao, Xiuli, & Zhu, Jinfu. 2007 (November). Grey programming for irregular flight scheduling. *Pages 1607 –1611 of: IEEE International Conference on Grey Systems and Intelligent Services, 2007. GSIS 2007.*

Zhao, Xiuli, Zhu, Jinfu, & Guo, M. 2007. Application of grey programming in irregular flight scheduling. *Pages 164–168 of: 2007 IEEE international conference on industrial engineering and engineering management*. New York: IEEE.

Zhu, G., Bard, J.F., & Yu, G. 2005. Disruption management for resource-constrained project scheduling. *Journal of the Operational Research Society*, **56**, 365–381.

Bibliography

An Introduction to Multi-Agent Systems .
Michael Wooldridge
2nd Edition, John Wiley & Sons Ltd, ISBN: 978-0-470-51946-2, 2009.

Distributed Intelligent Systems: A Coordination Perspective .
Abdellah Bedrouni, Ranjeev Mittu, Abdeslem Boukhtout and Jean Berger
Springer Dordrecht Heildelberg London New York, ISBN: 978-0-387-77701-6, e-ISBN: 978-0-387-77702-3, DOI 10.1007/978-0-387-77702-3, 2009.

Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence .
edited by Gerhard Weiss
The MIT Press, Cambridge, Massachusetts, ISBN: 0-262-23203-0, 1999.

Software Engineering .
Ian Sommerville
9th Edition, Pearson International Edition, ISBN: 978-0-13-705346-9, 2011.

UML 2.0 in a Nutshell .
Dan Pilone and Neil Pitman
O'Reilly ISBN-10: 0596007957 ISBN-13: 978-0596007959, 2005.

Glossary

Aircraft Recovery The process of assigning individual aircraft to a disrupted flight minimizing a specific objective (usually the cost) while complying with required rules.

Benevolence In a Multi-Agent System it is an *a priori* disposition for the agents to be helpful, e.g., to cooperate.

Bounded-Rationality The rationality of the agents are limited by the finite amount of time they have to make decisions and, possibly, by the limited information they have.

Crew Recovery The process of assigning individual crew members to a disrupted flight minimizing a specific objective (usually the cost) while complying with required rules.

Dead Head Crew See meaning of Extra-crew below.

Disruption An interruption to the regular flow or sequence of something.

Disrupted Aircraft An aircraft which cannot complete its original schedule.

Disrupted Crew member A crew member which cannot complete its original schedule.

Disrupted Flight A flight that, due to an unexpected event, cannot depart and/or arrive on its schedule time.

Disrupted Passenger A passenger that has lost one or more flight connections due to a disrupted flight.

Extra-Crew A crew member assigned to fly as a passenger on a specific flight so he can get to another city to work, where he will pick up his assigned trip sequence, or to return to is operational base after performing a flight.

Flight Recovery The process of repairing a flight schedule after a disruption, through specific actions like delay, cancel or divert flights from their original schedule, so that the flight delay is minimized.

GAIA An Agent Oriented Software Engineering methodology.

Hub-and-Spoke Network A system of air transportation in which local airports offer air transportation to a central airport where long-distance flights are available and vice versa.

Integrated Solution A solution that takes into consideration all the dimensions (or parts) of a problem simultaneously. In the specific case of the AOC a solution that includes the aircraft, crew and passenger dimensions of the problem.

Passenger Recovery The process of finding alternate itineraries, commencing at the disrupted passenger location and terminating at their destination or a location nearby, while minimizing a specific objective (usually the passenger trip time and the airline costs).

Rationality An agent is rational in the sense that it will try to maximize its individual utility and act according to its preferences. It is purely rational if it has full or perfect information about exactly what will occur due to any choice made and has the cognitive ability and time to weight every choice against every other choice.