FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Argumentation in the Resolution of Passenger Problems Using Mobile Devices

Jorge Filipe Monteiro Lima

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ana Paula Rocha

Second Supervisor: António Castro

July 14, 2016

Argumentation in the Resolution of Passenger Problems Using Mobile Devices

Jorge Filipe Monteiro Lima

Mestrado Integrado em Engenharia Informática e Computação

July 14, 2016

Abstract

In an airline company, the planning and subsequent monitoring of flights is a complex problem because it involves the consideration of multiple and costly resources and their dependencies. Unexpected events force a change in the previously envisaged plans, making it necessary for some entity to be responsible for the resolution of possible problems that occasional irregularities might cause.

The Operational Control Center (OCC) is the entity that manages the operations of an airline company in the moments that precede the realization of flights. Its primary objective is to solve eventual problems, minimizing as much as possible their impact in cost or delays. Three major consequences of these irregularities are the delays and cancellation of flights and the loss of transfers of passengers in transit to other destinations.

The resolution of an irregularity usually has three major dimensions: the plane resources, the passengers and the crew. If one of the affected parts is the passenger, currently he is informed by the company of an alternative route as being a consumed fact. If eventually he disagrees with the given solution he must head for the irregularities desk or call the company in order to find another solution.

This dissertation proposes a system where it is possible for an active participation of the passenger in the resolution of the problem through a mobile device, thereby trying to increase both the satisfaction and the commodity of the customer, minimizing the inherently existing drawbacks, by obtaining a more personalized solution.

The human passenger (hereinafter referred to simply as passenger) will exchange messages with a software agent (hereinafter PA) through a negotiation protocol based on arguments such that the resulting solution is according to his preferences. Through the use of Argumentation and based on the information the PA has according both to the passenger and the environment he is inserted, it is allowed, in addition to justify more verbally and clearly the choice of the current proposal, find the closest possible solution to the needs and criteria of each affected passenger with minimal cost to the airline.

Arguments composed by a claim and a reason may be used by the passenger in order to justify their decisions. In addition, the PA should also be able to understand the passenger's (human) arguments and formulate new arguments to rebut the received ones, thus presenting counter-proposals. Allied to these features, the possibility to use the mobile phone and the commodities that come along with it (traveling costs, length, time ..) makes it possible to carry out this process remotely without the need to look for the irregularities desk.

Resumo

Numa companhia aérea, o planeamento e consequente monitorização de voos é um problema complexo, pois implica a consideração de múltiplos e dispendiosos recursos e suas dependências. Eventos inesperados obrigam a alterações nos planos inicialmente previstos, pelo que se torna necessário existir uma entidade responsável pela resolução dos possíveis problemas que eventuais irregularidades provocam.

O Centro de Controlo Operacional (CCO) é a entidade que gere a operação de uma companhia aérea nos momentos que antecedem a realização dos voos, tendo como principal objetivo solucionar eventuais problemas minimizando tanto quanto possível o seu impacto em custo e atraso. Três grandes consequências destas irregularidades são os atrasos, cancelamentos dos voos e perdas de ligações dos passageiros em trânsito para outros destinos.

A resolução de uma irregularidade tem normalmente impacto em 3 dimensões: os recursos do avião, os passageiros, e a tripulação. No caso de o afetado ser o passageiro, este é geralmente informado pela companhia aérea do itinerário alternativo como sendo um fato consumado e, na eventualidade de não concordar com a solução dada, terá de se dirigir ao balcão de irregularidades ou telefonar, para tentar ver o seu inconveniente resolvido de outra forma.

Nesta dissertação é proposto um sistema onde seja possível a participação ativa do passageiro na resolução do problema, através de uma aplicação móvel, tentando deste modo aumentar a satisfação do mesmo, minimizando os inconvenientes inerentemente existentes, pela obtenção de uma solução personalizada.

O passageiro humano (doravante denominado simplesmente passageiro) irá trocar mensagens com um agente de software (doravante denominado PA) através de um protocolo de negociação baseado em argumentação de forma a que a solução resultante seja de acordo com as suas preferências. Através do uso de argumentação e baseado na informação que o PA possui relativamente quer ao passageiro quer ao ambiente que o rodeia é permitido, para além de justificar de forma mais verbal e clara a escolha da proposta atual, encontrar uma solução o mais próxima possível das necessidades e dos critérios de cada passageiro lesado com o mínimo custo para a companhia aérea.

Argumentos compostos por uma ambição e uma razão poderão ser usados pelo passageiro de forma a justificar as suas propostas. Para além disso, o PA deverá também ser capaz de entender os argumentos do passageiro (humano) e formular novos argumentos para rebater os recebidos, apresentando assim contra-propostas.

Aliado a estas funcionalidades, a possibilidade de uso do dispositivo móvel e das comodidades que deste advém (deslocações, morosidade, ..) torna possível a realização deste processo remotamente sem a necessidade de deslocação ao balcão de irregularidades.

Acknowledgements

First of all I would like to express my feelings of great gratitude towards my parents and sister without whom I would not have been able to achieve the things I have so far and whose love and guidance rose me up in all the bad moments I have came across.

Secondly, I would like to thank my girlfriend who stood by my side in every moment, and whose help and companionship kept me focused in the moments of higher stress. And also to my family for their understanding and support in all the moments I could not be present during the development of this dissertation.

Thirdly, I would also like to express my feelings of great respect and appreciation for both my supervisors, Professor Ana Paula Rocha and Professor António Castro for their endless support and help at any time or day. I wish you all the best, both personally and professionally.

Lastly, I would like to stress out the invaluable support I have received from my friends and colleagues, specially those from I003, for their company and help during this phase, and all my academic journey. With you everything is easier.

Finally, I would like to thank all the authors referenced throughout this dissertation. Without their hard work and commitment to science none of this work would be possible.

Jorge Lima

"The best way to predict the future is to invent it."

Alan Kay

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

ACC	Agent Communication Channel
ACL	Agent Communications Language
AMS	Agent Management System
API	Application Programming Interface
BDI	Belief-Desire-Intention
CSS	Cascading Style Sheets
DF	Directory Facilitator
ER	Entity-Relationship
FIPA	Foundation for Intelligent Physical Agents
GQN	Generic Q-Negotiation
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JADE	Java Agent DEvelopment Framework
JADEX	JADE eXtension
JS	JavaScript
JSON	JavaScript Object Notation
MAS	Multi-Agent System
MASDIMA	Multi-Agent System for Disruption Management
OCC	Operational Control Center
PA	Passenger Agent
REST	Representational State Transfer
URI	Uniform Resource Identifier
UUID	Universally unique identifier
XML	Extensible Markup Language

Chapter 1

Introduction

Nowadays, the presence of artificial intelligence in our daily life has come to a point where it can be found everywhere. Either in our search engines, or in automated kitchens, we can find pieces of code that help us go by our days, a little bit easily. The disruption management in airline operations control is a relevant domain where artificial intelligence can effectively help due to the environment's complexity, distribution and dynamism. This chapter will present the goals and motivations of this work, alongside with the structure of the thesis.

1.1 Context

In an airline company, one of the most important tasks is to control the operational plan, i. e., making sure that flights are executed according to the scheduled plan. When the normal functioning is affected by an unexpected event, disruption management appears in order to solve all possible issues. From aircraft to passenger, operation control centers have to effectively fix all the disrupted parts in the fastest time possible minimizing at the same time further costs.

1.2 Motivation and Objectives

When a disruption happens, it affects three parts or dimensions of the operational plan: aircraft, crew and passenger dimension. The passenger dimension is the last one considered in traditional problem resolution processes, leading to a minor range of possible good solutions. Since there is no interaction between the company and the disrupted passenger, the offered solution might not be optimal from the passenger point of view, having the need to search for the irregularities desk in order to have his problem solved. Nowadays, nearly 42% of the global population has access to a smartphone and internet, aiding users in their daily routines.

In this dissertation a novel system is introduced, allowing disrupted passengers to actively participate in the resolution process through the use of a mobile application. Argumentation techniques will be used in order to keep the flow of the discussion between the company and the passenger maintaining the proximity between both, at distance.

1.3 Dissertation Structure

Besides the introduction, this dissertation contains seven more chapters.

Chapter 2 contains a literature review on the topics of Airline Disruption Management, Multi-Agent Systems, argumentation-based negotiation and Mobile Application Development.

Chapter 3 gives an overview of the problem and outlines the general aspects of the proposed solution.

Chapter 4 defines the agent that will communicate with the disrupted passenger in behalf of the airline company along with the argumentation that will be used in the communication.

Chapter 5 defines and explains the server that will handle all the communication requests between the disrupted passenger and the agent.

Chapter 6 presents the used framework for the development of the mobile application and some of the designed interfaces.

Chapter 7 defines the scenarios used to test the system and presents the results obtained.

Finally in chapter 8 this dissertation is concluded by presenting the contributions that the current work has brought and some possible suggestions for future work.

In addition to these chapters this dissertation also includes two appendixes, one regarding the external data and other regarding the scenarios, both used to validate the system.

Chapter 2

Literature Review

2.1 Introduction

The objective of this chapter is to present an overview over the relevant literature to the problem and the proposed solution in order to understand and analyze the current state of the art. It is mostly focused in Airline Disruption Management, Multi-Agent Systems (MAS), Argumentation-Based Negotiation, and Mobile Applications Development.

2.2 Airline Disruption Management

In every airline company, disruption management is a topic that has to be thoroughly thought as it is crucial for the good functioning of the company, and all of its planned flights. In this context, a disruption can be defined as an unexpected event that might affect flights, causing departure or arrival delays or cancellations. These disruptions might be triggered by a large number of causes, such as environmental, mechanical and others. This process of management has several components and can be divided into the following [CRO14]:

- Aircraft Recovery

  The process of assigning a individual aircraft to a disrupted flight minimizing a specific objective (usually the cost and fight delay) while complying with the required rules.

- Crew Recovery

  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.

- 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.

- 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).

- Integrated Recovery

  A process that is able to recover all problem dimensions separately but not simultaneously.

Some approaches to these problems have already been implemented. However most of them lack some features regarding the passenger. The focus further on will be on the passenger recovery process.

2.2.1 Operational Control Center

Unexpected events force changes in the previous envisaged plans making it necessary for the existence of some entity responsible for the resolution of possible problems that occasional irregularities might cause.

Making good decisions is necessary in order to, besides solving the problem, increasing the customer's satisfaction and minimize the company's costs in the overall process.

Disruptions can affect the airline mostly in three dimensions:

- Resources
- Crew
- Passenger

The Operational Control Center (OCC) is the entity that manages the operations of an airline company in the moments that precede the realization of flights, having as the primary objective to solve eventual problems minimizing as much as possible their impact in cost and delays. Three major consequences of these irregularities are the delays and cancellation of flights and the loss of transfers of passengers in transit to other destinations.

According to Castro [dC08] the management of the operations, comprising activities of monitoring, event detection and problem resolution is, essentially, a manual process, heavily based on the tacit knowledge of the whole OCC staff. The act cycle of the OCC can be visualized in figure 2.1.

As stated before, the OCC deals with problems that affect three dimensions: Resources, Crew, and Passengers.

The focus in this dissertation will be about the Passenger dimension, and how to solve the, not yet solved, problems.

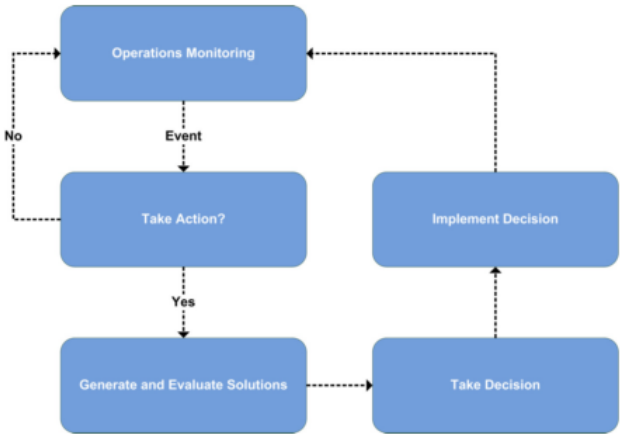


Figure 2.1: Act Cycle of the OCC [dC08]

2.2.2 Passenger Dimension

According to Castro [CRO14] a disrupted passenger is:

- A passenger that has lost one or more flight connections due to disrupted flight;
- A passenger whose itinerary contains a disrupted flight.

In the work of Bratu [BB06] two models were developed that optimize the balance between airline operating costs and passenger costs by identifying flight departure times and cancellation decisions. The first model, named Disrupted Passenger Metric, minimized the sum of operating and disrupted passenger costs. The second model, named Passenger Delay Metric, the delay costs are more accurately computed by explicitly modeling passenger disruptions, recovery options, and delay costs.

Another recent work on the passenger recovery problem was made by Zhang and Hansen [ZH08]. The authors introduce ground transportation modes as an alternative to the passenger recovery by air during disruptions in hub-and-spoke networks. An integer model with a nonlinear objective function allows to substitute flight legs with other forms of transportation, respecting the ground transportation times. The objectives of the model are aimed at minimizing passenger costs due to delay, cancellation or substitution, as well as minimizing the operating cost of the transportation.

These models however do not take into account the passengers personal interests. When offered a proposal the passenger might agree or disagree with it. In case of disagreement the disrupted passenger is obliged to go to the company's irregularities desk in order to see his problems solved in another way.

2.3 Multi-Agent Systems

Autonomous agents and multi-agent systems represent a new way of analyzing, designing and implementing complex software systems. The agent-based view offers a powerful set of tools,

techniques, and metaphors that have the potential to considerably improve the way in which people conceptualize and implement many types of software [JSW98]. A Multi-Agent Systems (MAS) is normally distributed as it is a software system composed of multiple interacting intelligent agents within some environment [CRO14]. Currently, agents have been used in a wide variety of applications, such as email filters to air-traffic control [JSW98]. Following the work of Jennings [JSW98] agents differ from autonomous systems because of the flexibility existent in agents, that those systems are not capable of. By flexible, it is meant:

- Responsive: agents should perceive their environment and respond in a timely fashion to changes that occur in it;
- Pro-active: agents should not simply act in response to their environment, they should be able to exhibit opportunistic, goal-directed behaviour and take the initiative where appropriate;
- Social: agents should be able to interact, when appropriate, with other artificial agents and humans in order to complete their own problem solving and to help others with their activities.

2.3.1 Development Frameworks

In order to build a system composed of several agents, some platforms already allow the development and deployment of one or several agents in some specified environment. The two most relevant frameworks will be described below.

2.3.1.1 JADE

JADE (Java Agent DEvelopment Framework) is a software framework to simplify the development of agent applications in compliance with the FIPA specifications for inter-operable intelligent multi-agent systems.

The Foundation of Intelligent Physical Agents (FIPA) is an international organization that is dedicated to promoting the industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent based applications [FIP].

According to Bellifemine [BPR99] JADE offers the following list of features to the agent programmer:

- FIPA-compliant Agent Platform, which includes the AMS (Agent Management System), the DF (Directory Facilitator), and the ACC (Agent Communication Channel). All these three agents are automatically activated at the agent platform start-up;
- Distributed agent platform. The agent platform can be split on several hosts (provided that there is no firewall between them). Only one Java application, and therefore only one Java Virtual Machine, is executed on each host. Agents are implemented as one Java thread and

Java events are used for effective and light-weight communication between agents on the same host. Parallel tasks can be still executed by one agent, and JADE schedules these tasks in a more efficient (and even simpler for the skilled programmer) way than the Java Virtual Machine does for threads;

- A number of FIPA-compliant DFs (Directory Facilitator) can be started at run time in order to implement multi-domain applications, where the notion of domain is a logical one as described in FIPA97 Part 1;
- Programming interface to simplify registration of agent services with one, or more, domains (i.e. DF);
- Transport mechanism and interface to send/receive messages to/from other agents;
- FIPA97-compliant IIOP protocol to connect different agent platforms;
- Light-weight transport of ACL messages inside the same agent platform, as messages are transferred encoded as Java objects, rather than strings, in order to avoid marshalling and unmarshalling procedures. When sender or receiver do not belong to the same platform, the message is automatically converted to /from the FIPA compliant string format. In this way, this conversion is hidden to the agent implementers that only need to deal with the same class of Java object;
- Library of FIPA interaction protocols ready to be used;
- Automatic registration of agents with the AMS;
- FIPA-compliant naming service: at start-up agents obtain their GUID (Globally Unique Identifier) from the platform;
- Graphical user interface to manage several agents and agent platforms from the same agent. The activity of each platform can be monitored and logged.

The communication between agents is made using the FIPA-ACL message specification. These messages are standardized to ensure interoperability and to provide well-defined process. They contains various parameters based on each situation. The most important are the performative (compulsory), sender, receiver and content.

JADE also uses the Behaviour abstraction to model the tasks that an agent is able to perform and each agent instantiate its behaviours according to its needs and capabilities [BPR99].

Behaviours can be used to perform any kind of actions, from receiving messages to data parsing.

The framework offers three types of primitive behaviours:

- SimpleBehaviour
  A basic class that can extended in various ways.

- CyclicBehaviour
  This behaviour stays active as long as its agent is alive and will be called repeatedly after every event.

- OneShotBehaviour
  Behaviour that is executed only once.

More complex behaviours exist or can be implemented by extending these primitive ones. Behaviours are triggered by the agent when it needs to perform some specific action.

2.3.1.2 Jadex

As stated previously, JADE platform focuses on implementing the FIPA reference model, providing the required communication infrastructure and platform services such as agent management, and a set of development and debugging tools. It intentionally leaves open much of the issues of internal agent concepts. The JADE eXtension Jadex is an implementation of a hybrid (reactive and deliberative) agent architecture for representing mental states in JADE agents following the BDI model [BLP03]. The mental state of an agent regarding the scenario of this dissertation is interesting because it would allow the representation of the agent interests more concretely.

2.3.2 Argumentation-Based Negotiation in Multi-Agent Systems

We can look at argumentation as a mechanism for achieving cooperation and agreement. According to Kraus [KSE98] the arguments used in this technique are utterances which aim is to change the intentions, beliefs and consequently actions of the listener. Within the context of a negotiating self-interest agent, this change of intentions could make agents more cooperative within the environment. Irrespective of which argument is used, the recipient of the message must evaluate the argument and decide whether or not to change its intentions and actions.

On its work Kraus [KSE98] describes argumentation as essential to bringing agreement in non-cooperative situations when agents have incomplete knowledge about each other or the environment. In these situations, agents receive information on each other via the exchange of messages.

When agents are non-collaborative, the process of argumentation is an iterative exchange of proposals towards the reduction of conflict and promoting achievement of the individual goals of the agents. In this scenario arguments are used as a means of dynamically changing the preferences, intentions, and actions of other in order to increase willingness to cooperate. Over repeated rounds, agents may analyze each other's patterns of behaviour to establish an analogy with the human notions of credibility and reputation therefore influencing the evaluation of arguments. By observing its arguer's reactions to the arguments, one can update and correct its model of the latter thus refining its planning and argumentation knowledge. The set of goals motivate the agent's planning process, meaning that the agent assigns different degrees of importance to different goals thus preferring to fulfill goals of higher importance [KSE98].

Compared to a regular negotiation, an argument-based negotiation allows agents to:

- Negotiation Stance Justification: An agent might have a compelling reason for adopting a particular negotiation stance. For example, a company may not be legally entitled to sell a particular type of product to a particular type of consumer or a particular item may be out of stock and the next delivery might not be until the following month. In such cases, the ability to provide the justification for its attitude towards a particular issue can allow the opponent to more fully appreciate an agent's constraints and behaviour.

- Persuasion to Change the Negotiation Stance: Agents sometimes need to actively change their opponents agreement space, or its rating over that space, in order for a deal to be possible. In such cases, agents seek to construct arguments which they believe will make their opponent look more favorably upon their proposal. Thus, arguments seek to identify opportunities for such change (e.g. a car salesman throws in a stereo with a car to increase the value of the good), create new opportunities for change (e.g. a car salesman adds a new dimension to the rating function by highlighting the cars novel security features) or modify existing assessment criteria (e.g. car salesman gets buyer to change evaluation function by convincing him that security is more important than mileage).

According to Jennings [JPNS98] in order to build argumentation-based negotiators there are some requirements that are described next:

- Mechanisms for passing proposals and their supporting arguments in a way that other agents understand.

- Techniques for generating proposals (counter-proposals or critiques) and for providing the supporting arguments; Proposal generation involves two main activities: (i) instantiating the negotiation object in accordance with the agent's acceptability region and its rating function; (ii) determining which argument(s) should accompany the agreement (if any) in order to maximise the likelihood of it being accepted. The complexity of the former point is determined by the nature of the strategic reasoning which is appropriate for the given negotiation protocol. This may vary from little reasoning, to maintaining complex models of negotiation opponents and trying to make predictions from them. In terms of the latter point, in the majority of cases there will be many types of argument which can be made in support of a proposal (varying from explanations to threats). In determining which ones to send, the agent needs to pick those arguments which are most likely to be effective, but within the constraints of the agent's negotiation objectives. Thus, for example, continually issuing threats may provoke short-term gains, but may not be a good long-term strategy if the agent has to interact frequently with the same group.

- Techniques for assessing proposals (counter-proposals or critiques) and their associated supporting arguments; Received proposals need to be evaluated to determine how the agent should respond. This evaluation involves two main facets: (i) assessing the desirability

of the proposal contained in the negotiation object; (ii) assessing the likely impact of the supporting arguments. From this, a number of potential outcomes are possible: the negotiation object is acceptable as it stands, the negotiation object alone is unacceptable but the supporting arguments overcome this and make the proposal acceptable, or the negotiation object is unacceptable and the supporting arguments are insufficient to warrant proposal acceptance4. Having assessed the proposal, the agent may decide to update its acceptability region or rating function to reflect the incoming proposal's arguments.

- Techniques for responding to proposals (counter-proposals or critiques) and their associated supporting arguments; Having assessed a proposal, the agent can respond by accepting it, by rejecting it, by generating a critique, or by returning a counter-proposal. So the first functional requirement is to determine which of these courses of action should be taken. In the case of a critique, the agent has to determine what components it wants to accept and which it wants to reject, which issues it intends to provide constraints on, and what such constraints should be. It must then decide what arguments (if any) it will offer in support of this stance, and how it should respond to any arguments which accompanied the incoming proposal (varying from ignoring them to trying to undermine them). Counter-proposals are handled in a broadly similar manner, except that rather than giving feedback and constraints the agent has to instantiate the negotiation object with particular values. While these argumentation specific capabilities undoubtedly increase the complexity of the agent, we feel such efforts are justified by the increased rewards which argumentation-based negotiation promises.

Some examples of argumentation types are as follows (increasing in strength) [KSE98]:

- Appeal to prevailing practices
- A Counter-Example
- Appeal to a Past Promise
- Appeal to Self-Interest
- A Promise of Future Reward
- A Threat

2.3.3 MASDIMA

MASDIMA or Multi-Agent System for Disruption Management is a system that aims to manage the operation of airlines, monitoring unexpected events that may affect and cause flight delays. As referred previously these events can affect the aircraft, crews and passengers and MASDIMA analyzes the impact on these three dimensions, looking for the best integrated solution and complying with the time available for arriving to a solution [MAS].

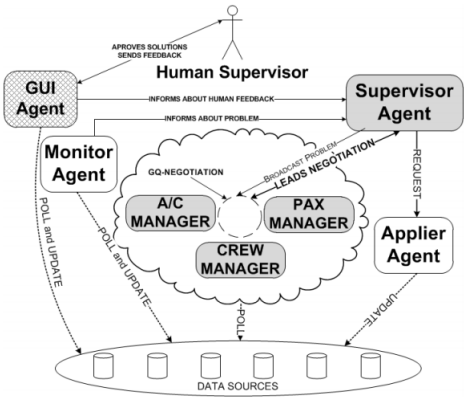


Figure 2.2: MASDIMA Multi-agent System Architecture [CRO14]

MASDIMA uses software agents in a distributed environment. Each software agent (manager) represents a part of the problem including its preference and goals: aircraft, crew and passenger. Another software agent (supervisor) represents the global view of the airline company. Through an automated negotiation process, called Generic Q-Negotiation (GQN), the best integrated solution is chosen according to the global interest of the airline. This results in an autonomous, automated and adaptive system that also includes the human agent in the decision process.

Despite all the benefits provided for the company, MASDIMA lacks a personalized solution for each disrupted passenger, providing only a single solution for its problem. The current work would fill this gap and enhance MASDIMA functionalities, by providing a personalized solution to the disrupted passenger where himself can participate.

2.4 Mobile Applications Development

With over ten billion mobile Internet devices expected to be in use in 2016, it is predicted that the mobile application industry will grow tremendously to match demand and keep up with ever-evolving technologies.

2.4.1 Development Frameworks

The number of frameworks that allow mobile application development has been increasing in the past few years, varying from programming language to target device. In the next sub-sections two frameworks will be introduced and compared to provide more information about them. Both these frameworks deploy for multi-platform devices, meaning that regardless of the programming language used, applications can be deployed to all target platforms (Android, iOs, ...).

2.4.1.1 Ionic Framework

The Ionic Framework appeared in 2013 with a new concept of mobile development. It takes advantage of technologies such as Angular JS and Cordova to deploy mobile applications using

Table 2.1: Ionic and Codename One Comparison

Framework	Used Language	Platforms	Cordova	Debugging	Build
Ionic	HTMl, JS and CSS	Android, iOS(requires a Mac), Windows Phone, Web	Yes	Browser and Console	Local
Codename One	Java	All platforms	Can be used, but not obligatory for accessing hardware	Same as Java applications	Cloud Build Server

Web programming languages such as HTML, CSS, and Javascript [ION].

Cordova wraps the HTML/JavaScript application into a native container which can access the device functions of several platforms allowing the use of native features, such as camera, gps, and more. Using this framework one can build a mobile application from scratch as if it was developing a web application.

2.4.1.2 Codename One

Codename One resembles to Ionic only in the way that both are cross-platform. Unlike Ionic, Codename One uses only java as a programming language. Java is familar to tens of millions of developers world wide guaranteeing code maintainability well into the future.

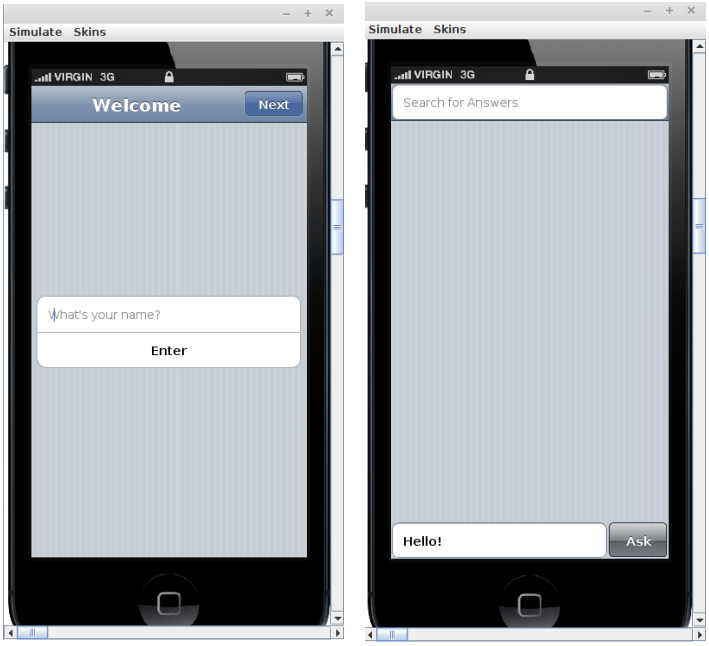
What Codename One does is translating all the code to native code or the native VM resulting in performance that matches the performance of native code, achievement that other frameworks cannot match.

Taking advantage of the programming language developers can use the rich tools available for them such as Eclipse/NetBeans/Intellij to work with the code. Codename One has its own GUI builder and many tools to track issues in the code. The biggest benefit though is in the build server which generates a native application for user without having to deal with all of the complexities of building a native app for every platform. [CON]

2.4.2 Experiments

Some experiments were done regarding the Codename One framework, and since there was previous experience with the Ionic Framework no experiments were performed regarding the latter. A simple application was developed to test the ease of use and the performance of Codename One and is described in the images 2.3a and 2.3b. Previous knowledge in Java made it possible to develop this application in just a couple of hours.

A list of overall results is displayed in table 2.1.



(a) Initial Screen **(b)** Input Screen

Figure 2.3: Application Screenshots

2.4.3 Related Mobile Applications

In this section a list of related applications regarding the airline industry will be introduced highlighting its main features.

- American Airlines [AA]
  - Check In
  - Mobile Boarding Pass
  - Seat Changing
  - Flight Schedules
- Turkish Airlines [TA]
  - Mobile Booking and Reservation
  - Check In
  - Arrival/Departure Information
  - Flight Schedules
- United Airlines [UA]
  - Mobile Booking
  - Check In

- Mobile Boarding Pass
- Flight Status
- Flight Reminders
- Seat Changing

None of the provided Airline applications has the possibility of using user input to help in any task, such us providing a interface that allows direct client communication with the company actively participating in the solution of problems, such as delays and flight cancellations.

2.5 Summary

This chapter provided a literature review on the topics of Airline Disruption Management, Multi-Agent Systems, Argumentation-Based Negotiation and Mobile Application Development. The review covered aspects related to the Passenger problem when applied in an airline company, and a general overview on multi-agent systems and argumentation based negotiation. Some experiments performed were also described in this chapter, including comparisons between platforms and frameworks, related to agent development and mobile applications development. Finally, it was concluded that the lack of personalization in the passenger dimension resolution process approach was a real problem that affected millions of passengers annually, making the purpose of this dissertation highly valuable both the company and the disrupted passenger.

Chapter 3

Preliminaries

The purpose of this chapter is to provide an overview of the problem and a brief introduction to the proposed solution that will be explained in detail throughout the subsequent chapters.

3.1 Passenger Problem

The resolution of the passenger problem has, as stated previously, many solution approaches. The drawback found in the existing solutions is that they are not complete as they do not include the passenger in the attempt of resolution. By this it is meant that there is no active participation of the disrupted passenger in the problem resolution. Instead, if the disrupted passenger disagrees with the airline company's decision, he would have to reach their offices in person and try to change the alternative provided by the company. As stated by Maher [Mah15] the passenger recovery is generally considered as the final stage in the resolution process, and hence passengers experience unnecessarily large impacts resulting from the referred disruptions making this approaches far from optimal in the disrupted passenger point-of-view. The lack of a personalized solution might decrease the customer satisfaction and therefore the loyalty to the airline company.

3.2 Overview of the proposed solution

In this section a novel approach to the resolution of the passenger dimension problem is described. Taking advantage of the current technologies in areas like Artificial Intelligence, Multi-Agent Systems and Mobile Application development, a system will be defined, allowing the disrupted passengers to actively participate in the resolution of the problem. A software agent (hereinafter PA) alongside with a mobile application will allow the human passenger (hereinafter referred to simply as passenger) to refute or agree with proposals made by the company. The refusal or agreement will be paired with an argument in order to justify the decision. Both the agent and the passenger will trade messages resulting in an optimal solution for both the passenger and the company. The system architecture can be represented as shown in figure 3.1.

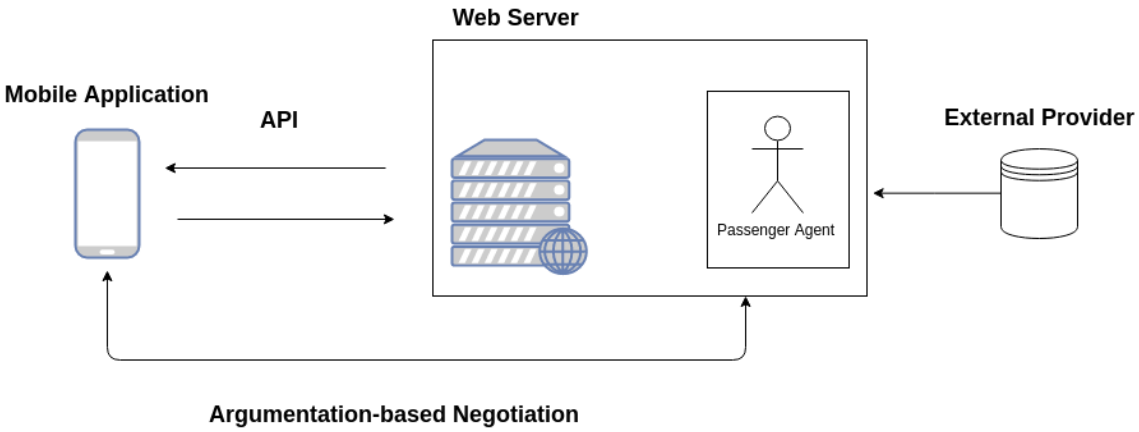


Figure 3.1: System Architecture

A brief introduction of each element of the system will be introduced next except for the external provider that as the name says is external to the system.

3.2.1 The Passenger Agent

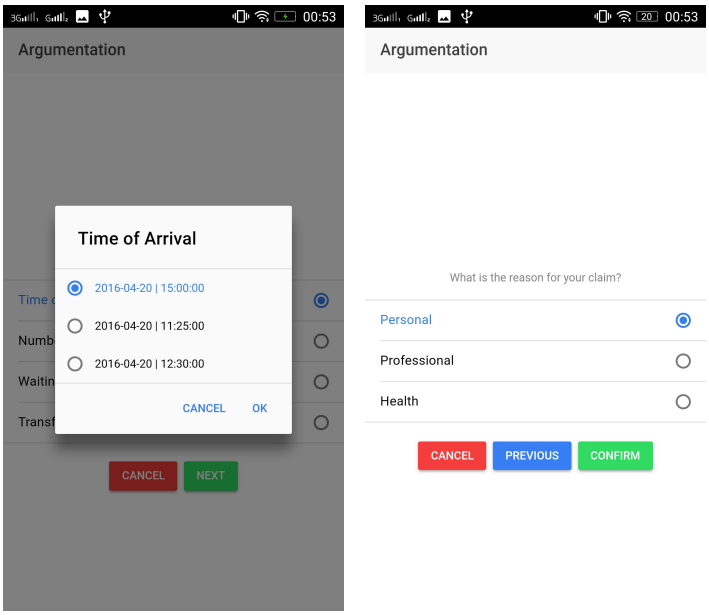
In order to ensure proper communication between the passenger and the company without the need of human supervision, a software agent (the PA) will be defined and implemented. The PA will continuously offer new alternatives in case of disagreement with the passenger if the latter presents valid arguments. The passenger will also communicate with the agent, agreeing or disagreeing with it through the use of argumentation techniques. This agent will be implemented using the JADE framework referred previously and a detailed description of this component is presented in chapter 4.

3.2.2 Argumentation-Based Negotiation

As stated previously, the use of argumentation will allow a verbose justification for each proposal and counter-proposal made by each participant, either the passenger or the PA. A claim and a reason will compose an argument that will be exchanged along with the rejection of alternatives, allowing the passenger to engage on an argumentation with the PA. An example is shown in figure 3.2.

3.2.3 The Web Server

To ensure proper communication channels between the application and the PA a web server will be developed. The web server will wrap an agent container which contains the PA. This way it will be able to exchange messages through the exposure of an API. After deciding which message to send, it will be converted to an API request according to its characteristics, providing an adequate communication environment.



(a) Claim Interface **(b)** Reason Interface

Figure 3.2: Argumentation Screenshots

3.2.4 The Mobile Application

Through the use of a mobile application the passenger will be able to start a discussion with the company regarding its problem without the need of dislocations or any kind of effort besides some clicks on his smartphone. This application will present the passenger with a simple yet intuitive interface allowing the exchange of messages. The passenger will be presented with arguments which options and variables he can change according to his opinion.

3.3 Summary

In this chapter an overview of the problem in hands was presented, followed by the proposed solution. The full system architecture was introduced, including the software agent, the mobile application, argumentation issues, and finally the wrapping server. These components will be described in detail in the following chapters.

Chapter 4

Passenger Problem Resolution: the Passenger Agent

In the last chapter the major problem and the proposed solution were presented. In this chapter information regarding the first two elements of the solution, the PA and the argumentation process, will be presented.

4.1 Passenger and Gateway Agents

In this scenario the PA is one of the most important pieces since it's the one who will represent the airline company while communicating with the disrupted passenger.

To better understand how this agent works this section will present some important concepts on software agents and how they were used in our advantage.

In the proposed system it was decided to have two software agents:

- Gateway Agent
- Passenger Agent

The gateway agent is a simple mediator agent, that allows the communication between the MAS and the server that will be explained further on. Its only function is to wait for messages and send them to the correct agent. A detailed diagram is shown in figure 4.1

The passenger agent (PA) is the one that will be assigned to each disrupted passenger in order to try to solve his problem.

4.1.1 Behaviours

As stated in 2.3.1.1 Jade uses a **behaviour** abstraction that allows the developer to personalize the agent to suit his needs. Taking advantage of this, the PA was equipped with four main behaviours:

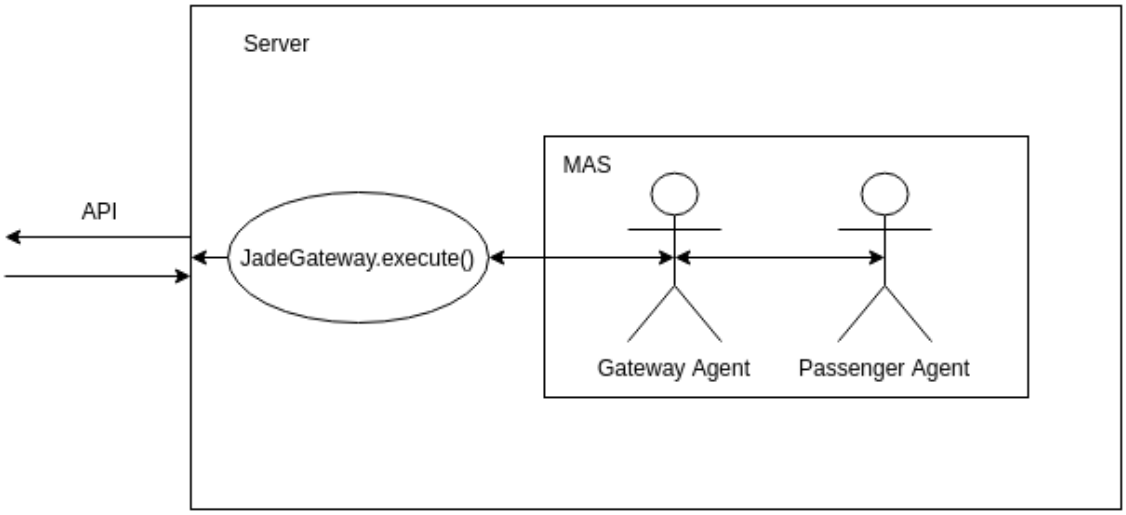


Figure 4.1: Gateway Agent and Passenger Agent interaction

- ConnectionBehaviour
- ProposeBehaviour
- ArgumentBehaviour
- RevertClaimBehaviour

Since these behaviours represent single actions, each one extends the OneShotBehaviour referred in 2.3.1.1. The PA also makes use of a single cyclic behaviour to wait for messages from the gateway agent. After evaluating the received message the PA decides which of the main behaviours it should launch to achieve the desired effect. These are described in next paragraphs.

Connection Behaviour

As stated previously this behaviour extends the OneShotBehaviour meaning that it will run once and then end. The PA will launch this behaviour when an intent of connection is sent by the passenger. It then verifies if there is any disruption affecting that passenger and in a positive case sends him the disruption details finishing afterwards.

Propose Behaviour

This behaviour is triggered when the passenger intends to start the negotiation process and sends back to the latter the first four alternatives with the least cost for the airline company. No reasoning is made while obtaining these first alternatives since the passenger has not yet argued.

Argument Behaviour

It's in this behaviour that the PA analyses the passenger arguments and decides whether or not to send new alternatives. To achieve such purpose an algorithm was designed and will be demonstrated in chapter 4.2.

Revert Claim Behaviour

This behaviour allows the passenger to void an already sent claim, sending back the best proposals that match the desired claims.

4.1.2 Messages

Jade takes advantage of the FIPA-ACL specification to deal with the messages exchanged between agents, in this case the PA and the gateway agent. Each FIPA-ACL message is composed by several parameters, but in this scope only the performative, that identifies the type of message, the sender and receiver that are the participants of the message exchange, and the content that refers to the content to be exchanged, were used. The FIPA specification provides performatives for most cases of a communicative act.

When the gateway agent receives a message from the server, it translates it into an ACL message defining each performative according to the message received.

The following table refers to the connection between the received messages from the server and the correspondent ACL message:

Table 4.1: Correspondence between received messages from API and ACLMessage

From Server (Gateway Agent)	ACLMessage (Passenger Agent)	
	Performative	Content
Connection Intent	INFORM	Connection message
Request Alternatives Intent	CFP	Request message
Reject Alternatives Intent	REJECT-PROPOSAL	Arguments message
Accept Alternative Intent	ACCEPT-PROPOSAL	Selected alternative
Revert Claims Intent	REQUEST	Claims to remove
Retrieve Current Claims Intent	REQUEST	Current Claims

A representative diagram is presented in figure 4.2.

4.2 Argumentation

It's through argumentation that the PA and the passenger will negotiate and stand by their point of view until a mutually acceptable conclusion is reached. The PA will start by presenting the passenger with four initial alternatives to his disrupted flight. After analyzing those alternatives the passenger should decide whether or not to start an argumentation process with the PA by rejecting the presented proposals. In case of acceptance, the negotiation process terminates successfully.

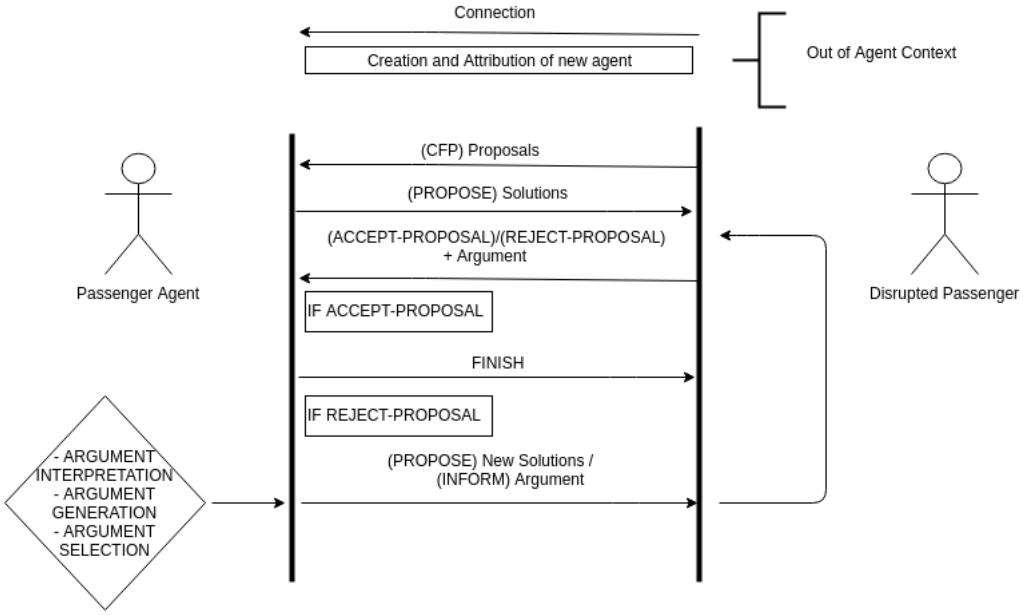


Figure 4.2: Message Flow: Passenger Agent <-> Disrupted Passenger

It's important to point out that this process can end up unsuccessfully meaning that no agreement was achieved and the disrupted passenger will have to use the traditional methods to change the final outcome, if he so wishes.

Before starting with the argumentation itself the argument structure had to be defined, and so it was decided to define it as a pair A = <Claim, Reason>. The argument is therefore composed by two elements, a claim and a reason. In this scope, the claim is how and what the disrupted passenger wishes to improve in the previously analyzed alternatives. For instance, time of arrival being before some date or number of transfers being inferior to some number. The reason element is why the passenger thinks his claims should be approved. It's this reason that should be convincing or persuading enough to change others mental state.

In this particular case, the disrupted passenger will argument about parameters such as those referred above, and the PA will argument on the rejection of the latter.

Now that the argument was defined, and as stated in 4.1.1 it was necessary to develop an algorithm that could analyze, in case of argumentation, the arguments received by the disrupted passenger.

The argumentation reasoning by the disrupted passenger will be made solely by himself so the focus in the argumentation process will be on the PA reasoning. The claims and reasons the disrupted passenger can use are limited, since natural language processing is not part of this dissertation scope.

The defined Claims are as follows:

- Time of Arrival
- Waiting Time on Transfers

- Transfer Location
- Number of Transfers

And the Reasons are:

- Personal
- Professional
- Health

Each of the latter elements, when combined form an **argument**.

When the disrupted passenger decides to start a negotiation process, a message containing his argument is sent to the server, redirecting this message to the PA assigned to that passenger. This PA will trigger an argument behaviour as referred in 4.1.1 and start reasoning about the presented scenario. This scenario will not only contain the arguments the disrupted passenger has sent, but also knowledge the PA might have about the latter. The argumentation algorithm is represented as follows:

Algorithm 1: Argumentation Algorithm

Input: Argument
Output: Response

```
1  best = null;
2  if argumentVeracity >= veracityLimit then
3      previousAlternatives = checkPreviousAlternatives();
4      if previousAlternatives not empty then
5          best += previousAlternatives;
6      end
7      args = findAttackingArguments();
8      if args is empty then
9          for Round in roundsSoFar do
10             currentClaims = getCurrentClaims();
11             bAlternatives = findBestAlternatives(currentClaims);
12             best += bAlternatives;
13             Response = best;
14         end
15     else
16         Response = args;
17     end
18 else
19     Response = error message;
20 end
```

At first, the PA verifies the arguments veracity by calculating how many times the passenger has sent that argument in that context. If that value is higher than a minimum limit, it then proceeds with the negotiation, rejecting otherwise. Next the PA will check for previous alternatives that match the claims sent so far and add them, if available, to the output array (attack by rebut). Next it will verify if any argument that attacks the received one can be found , returning it if found (attack by undercut). If none of the latter steps is triggered the PA can now move on and find new alternatives that match the desired claims before sending them back to the passenger.

Check Previously Sent Alternatives

This method will verify from all the already sent alternatives all that still match the passenger requirements. The designed algorithm is as follows:

Algorithm 2: Check Previously Sent Alternatives Algorithm

```
   Input: Argument
   Output: Response
 1 matches = [];
 2 if numberOfRounds > 1 then
 3     for Round in roundsSoFar do
 4         for Alternative in AlternativesSentInRound do
 5             for Argument in ArgumentsToClaim do
 6                 if alternativeMatchClaim then
 7                     matches += true;
 8                 else
 9                     matches += false;
10                 end
11             end
12             if matches not contains false then
13                 Response += Alternative;
14             end
15         end
16     end
17 end
```

Finding Attacking Arguments

This method will verify if the PA can refute the passenger argument by attacking it. The PA will search in available data for elements that could verify the passenger argument. For instance, searching in the passengers special requests for health and/or professional related requests. The

passenger importance is a relevant variable since it will define the range of acceptability of the argument as the PA will soften its aggressiveness based on that importance.

Algorithm 3: Find Attacking Arguments Algorithm

```
   Input: Argument
   Output: Response
 1 paxIntel = retrievePaxHistory();
 2 args = verifyIntel(paxIntel);
 3 if args not null then
 4 |   Response = null;
 5 else
 6 |   paxImportance = paxIntel.importance;
 7 |   if paxImportance > importanceLimit then
 8 |   |   Result = null;
 9 |   else
10 |   |   Result = args;
11 |   end
12 end
```

Find Best Alternatives

In this part the PA will find all the possible alternatives that match the current claims, and add them to the output array in order to send them to the passenger. The alternatives are ordered in crescent order of cost for the company, so the first ones retrieved are the ones with less cost, minimizing the loss of the company at each step.

After finishing its reasoning PA will whether send to the disrupted passenger new alternatives, or a message containing the rejection message and a new round will start.

4.2.1 Reverting Claims

During the negotiation process the disrupted passenger might have the need to rollback some claim in order to see his preferences matched. In this case he could use the revert claim feature that would remove the selected claim from the claims used so far, allowing the disrupted passenger to review his selections in each round of the negotiation.

4.3 Summary

This chapter can be divided into two parts, one about the PA, its definition and the exchanged messages between the latter and the disrupted passenger, and the second part about the argumentation part of the negotiation process.

The first one started by defining the PA and describing the behaviours that represent its actions. It were also defined the relations between the messages received in the server and the ACLMessage the PA will receive.

In the second part the argumentation is introduced in this context and it started by defining the argument structure to be used by each participant along with the possible components of that argument. The algorithms that guide the PA through the argumentative reasoning were also described in this part.

To end this chapter the concept of claim reversal was also introduced.

The following chapters will continue with the description of the rest of the system components.

Chapter 5

Back-end Architecture

The communication between the disrupted passenger and the MAS couldn't be performed directly so a back-end structure had to be defined containing the server that would mediate all the communications. In this chapter the back-end architecture will be described along with the explanation of how it puts everything together.

5.1 Server

For the communication between the passenger agent and the disrupted passenger to be performed a server was implemented using Jetty a Java HTTP (Web) server and Java Servlet container [Jet]. This server will receive messages from the disrupted passenger and send them to the gateway agent that will afterwards transmit back the response received from the PA.

For this to be possible the server provides web services that can be accessed by the disrupted passenger to communicate with the agent.

Those web services were implemented using a REST architectural pattern. The representational state transfer (REST) is an architectural style consisting of a coordinated set of components, connectors, and data elements within a distributed hypermedia system, where the focus is on component roles and a specific set of interactions between data elements rather than implementation details. Its purpose is to induce performance, scalability, simplicity, modifiability, visibility, portability, and reliability [Fie00].

Java was the programming language used to implement the server and it already provided the JAX-RS API for the web services creation. JAX-RS or Java API for RESTful Web Services is a Java programming language API that provides support in creating web services according to the Representational State Transfer (REST) architectural pattern. JAX-RS uses annotations to simplify the development and deployment of web service clients and endpoints[Jax]

5.1.1 Endpoints

As stated previously by using the REST pattern the server needed to expose an API allowing the passenger to perform requests on the server. The designed endpoints can be described as follows:

- Connect

  Starts the communication and instantiates a new agent for this passenger.

Endpoint	HTTP Method	URI	Parameter	Content	Media Type
Connect	POST	http://../api/connect/	User UUID	Message	JSON

- CFP

  Sends a call for proposals. Four initial alternatives will be sent.

Endpoint	HTTP Method	URI	Parameter	Content	Media Type
CFP	POST	http://../api/cfp/	User UUID	Message	JSON

- Accept Proposal

  Sent when the user accepts one proposal and finishes the negotiation.

Endpoint	HTTP Method	URI	Parameter	Content	Media Type
Accept Proposal	POST	http://../api/acceptproposal/	User UUID	Message	JSON

- Reject Proposal

  Sent when the user rejects the proposals and arguments about the previous ones.

Endpoint	HTTP Method	URI	Parameter	Content	Media Type
Reject Proposal	POST	http://../api/rejectproposal/	User UUID	Arguments	JSON

- Revert Claim

  Sent when the user decides to revert a claim, meaning that that claim will no longer be included on the problem solution.

Endpoint	HTTP Method	URI	Parameter	Content	Media Type
Reject Claim	POST	http://../api/revertclaim/	User UUID	Claim	JSON

- Get Current Claims

  Sent when the verification of the current claims in this negotiation is needed.

Endpoint	HTTP Method	URI	Parameter	Content	Media Type
Get Current Claims	GET	http://../api/currentclaims/	User UUID	-	JSON

These endpoints will be the entry point for the communication between the disrupted passenger and the PA.

5.1.2 Communicating with the Agents

As referred in 4.1 there was the need to implement some kind of mediator that facilitated the exchange of messages between the server itself and the embed MAS. For that purpose Jade provides a GatewayAgent class. By extending the gateway agent to the latter class we could access it by using the JadeGateway.execute() method as shown in 4.1. By processing the command received by this method the gateway agent decides what kind of ACLMessage the command translates to, and then sends it to the PA.

5.2 Database

The passenger negotiation history is one of the most important elements of the argumentation process since the PA will reason not only about the passenger choices on the application, but also about his history and preferences.

For that matter the development of a database containing the passengers history was needed.

MongoDB is a cross-platform, non-relational, document oriented database that provides high performance, high availability, and easy scalability. MongoDB works on concept of collection and document and for this reasons was the first choice for the database implementation [Mon].

Each passenger document contained his basic information such as given name, surname and an unique identifier. It also contained information about the passenger preferences, such as seat location. Finally, information about all the previous negotiations the passenger might have had with the airline company and his importance were also stored.

A visual representation in form of a ER diagram is shown in figure 5.1

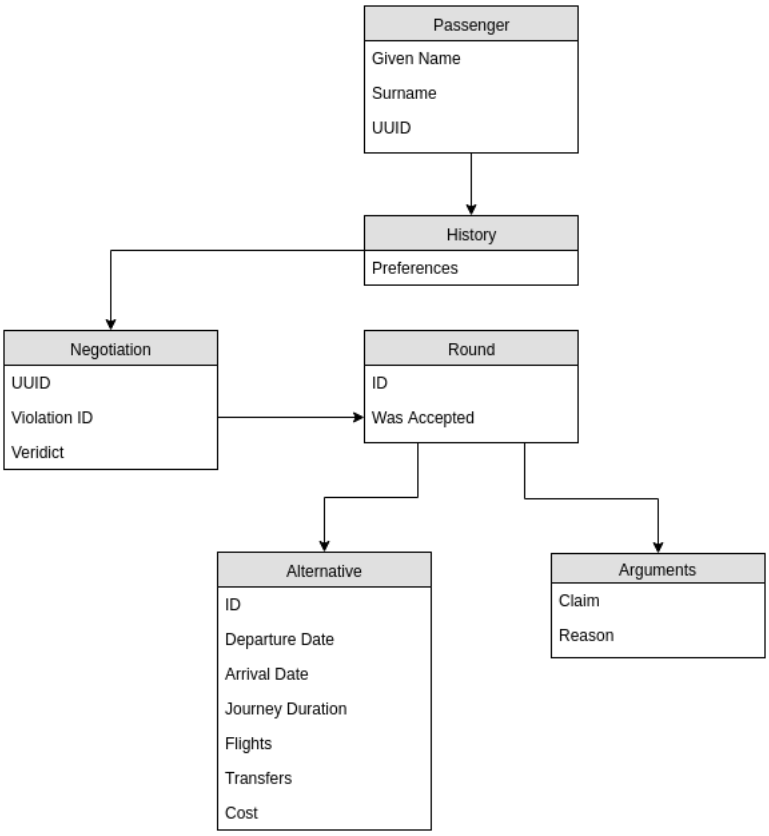


Figure 5.1: ER Diagram of the Database

5.3 External Data

For the approach to be realistic, real data was used for testing purposes. Information about flight disruptions, alternative flights, and disrupted passengers in the period of time from 2016-04-19 at 15:35:56 to 2016-04-19 at 16:11:17 was kindly provided by an European airline company and used while testing the system.

This information was provided as XML files and refer to the following data:

- Problems

  This file contained information about a problem that caused a disruption.

- Problem Violations

  This file contained information about a violation caused by a problem. It provided information about the affected resource, the type of violation, the flight number and other parameters.

- Passenger List

  This file contained information about each passenger of a disrupted flight.

- Alternatives

  This file contained information about the alternatives to a specified flight.

With this information it was possible to test the system with a more realistic approach and obtain better results.

5.4 Summary

This chapter introduced the back-end components that would facilitate the exchange of messages and data retrieval in the system.

It started by defining the server and its endpoints, along with an explanation of how the communication between the PA and the disrupted passenger would be performed.

Since there was a need the implement a database, this chapter also described the used database and its components.

To conclude this chapter the external data used in the system experiments was also described.

The next chapter will introduce the mobile application that reads the disrupted passenger inputs.

Chapter 6

Front-end Application

Having all the back-end structure set, the system was missing a way for the disrupted passenger to interact with the PA, and so a mobile application was developed. In this chapter the used framework for the development of the mobile application and the interfaces will be presented.

6.1 Framework

As stated in chapter 2 there were many possibilities to choose from regarding the mobile application development and some studies were made regarding each of the frameworks referred in the chapter 2.4.

6.1.1 Codename One

At first Codename One seemed the most appealing framework since it just uses Java as programming language. After experimenting with it for a few weeks it was then realized that it was not so adequate for the type of application in mind.

6.1.2 Ionic and Ionic2

Ionic was the next on the framework list, and previous knowledge on this framework allowed for the recreation of the previous developed application in Ionic rapidly.

Albeit still being in beta version an application in Ionic2 was also developed since it uses newer technologies that were not used in Ionic. The good thing about Ionic2 is that it modularizes each component of the application as if everything was a component allowing better programming patterns. Some interface examples will be presented next.

6.2 Interfaces

It was not in the scope of this dissertation to develop a production admissible application however great effort was put in the design and structure of the developed one.

After starting the simulation the disrupted passenger is presented with four alternatives to his disrupted flight as shown in figure 6.1. In this screen the disrupted passenger can choose one from four alternatives, accept, reject all, see more information regarding each alternative, and revert a claim or leave the process.

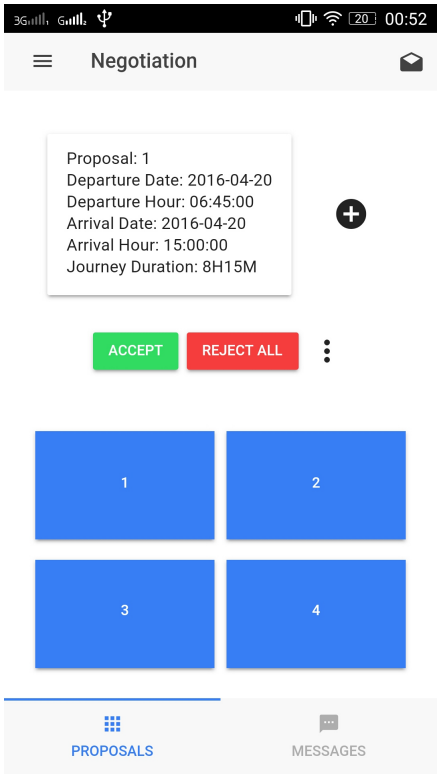
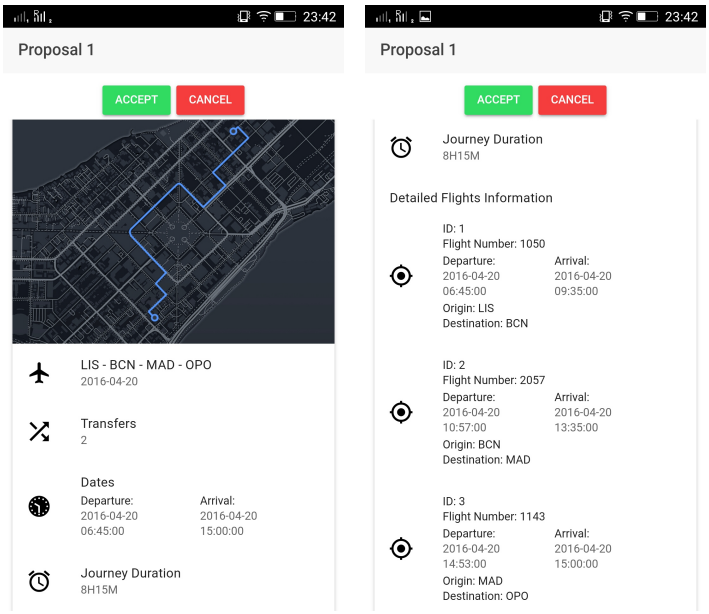


Figure 6.1: Main Window

When the plus sign is clicked, a new window appears containing detailed information regarding each alternative. An example is shown in figure 6.2a and 6.2b. This window contains several information about each alternative, such as the full route, the number of transfers, departure and arrival dates, journey duration, and detailed information about each segment flight of this alternative. A segment flight is considered to be a flight needed to fulfill some route. In the example shown, the three presented flights are segment flights. It is also possible to accept or reject the selected alternative in this window.

In the figures 3.2 in chapter 3, the interfaces of the argumentation part were shown. This windows appear when the disrupted passenger decides to reject all alternatives. When that event is triggered, a window containing a slide show appears where each slide represents a phase in the argumentation process.

The first slide refers to the claim where the disrupted passenger is presented with four possible claims from which he can select one. He can also choose a reference value from which he wants to improve the alternatives. For instance, selecting claim time of arrival and a reference value of 2016-19-04 - 22:20 would lead to alternatives where the time of arrival would be inferior to the provided reference value.



(a) More Info Window 1 **(b)** More Info Window 2

Figure 6.2: More Information Windows

In the second slide the disrupted passenger might choose from personal, professional and health reasons the one that matches his.

The passenger is obliged to provide values for all inputs before sending the argumentation message to the server by clicking in the confirm button in the last slide.

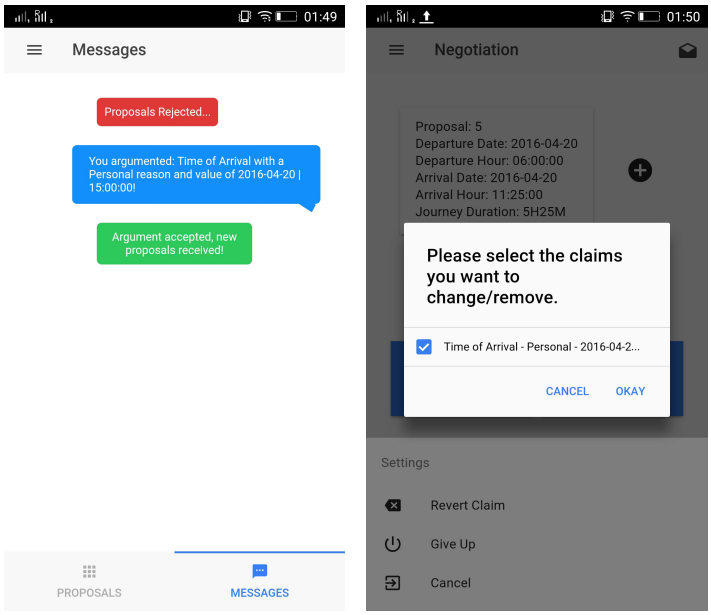
Figure 6.3a shows the messages window where the disrupted passenger might review the sent or received messages and also approval or rejection of arguments.

If the disrupted passenger wishes to remove a claim he might do so by clicking in the three dots in the main window and select the revert claim option. As shown in figure 6.3b it is presented a selection box from where the passenger might choose a claim he has already sent. By proceeding with this action the disrupted passenger will send a revert claim message containing the claims he wishes to remove from the negotiation process.

With the features referred above the disrupted passenger can successfully negotiate and argument with the passenger agent.

6.3 Summary

This chapter introduced the used framework to develop the mobile application, in this case Ionic2 along with a brief description of the experience with each of the possible ones. It continues by presenting most of the created interfaces and the information they provide followed by a detailed explanation of what it allows the disrupted passenger to perform while using it.



(a) Messages Window **(b)** Revert Claim Selection

Figure 6.3: Messages and Revert

Chapter 7

Experiments and Results

Now that the system is completely defined some experiments should be performed in order to validate and test it. As referred in the chapter 5.3 an European airline company kindly provided information regarding disruptions, passengers and alternatives that made the testing be as realistic as possible. For that three scenarios were created. Each scenario reflected a disruption, a passenger and several alternatives. This chapter will start by introducing the different scenarios used to validate the system, followed by the analysis of the results obtained.

7.1 Scenarios

Each of the following scenarios was created in the attempt to verify not only how people (passengers) would react in specific situations, but also to verify how the PA reacts under different circumstances. The experiments used real data and each scenario contains information about the disrupted flight, the disrupted passenger, and other relevant elements in the period of time from 2016-04-19 at 15:35:56 to 2016-04-19 at 16:11:17. It's important to point out that the passenger history was fictionally created since that information was not available.

For testing purposes ten subjects interacted with the application in each of the scenarios, in order to improve the initial alternative received by the airline company according to their preferences.

7.1.1 Scenario 1

In the first scenario the disrupted passenger is supposedly traveling from Amsterdam to Porto, having as segment flights Amsterdam->Lisbon (AMS-LIS) and Lisbon->Porto (LIS-OPO) but the flight AMS-LIS suffered an eighty minutes delay, causing the passenger to miss the transfer from Lisbon to Porto as in figure 7.1.

The flight the passenger had missed had the following details:

- Flight Number: 1958
- Delay: 80

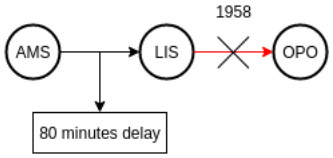


Figure 7.1: Scenario 1

- Arrival: 19-04-2016 22:20

Relevant information about the passenger was also provided, and in this case referred that this passenger had used in two previous negotiations arguments with a Personal reason one time in each negotiation.

The airline provided one initial alternative from a total of twenty two:

- Route: Lisbon->Barcelona->Madrid->Porto
- Departure: 20-04-2016 06:45
- Arrival: 20-04-2016 15:00
- Journey Duration: 8h15m

It is then said that the passenger wishes to change this alternative according to his preferences by interacting with the provided application.

7.1.2 Scenario 2

In the second scenario the disrupted passenger was supposedly traveling from Frankfurt to Madeira, having as segment flights Frankfurt->Lisbon (FRA-LIS) and Lisbon->Madeira (LIS-FNC) but the flight FRA-LIS had suffered an forty-two minutes delay, causing the passenger to miss the transfer from Lisbon to Madeira as in figure 7.2.

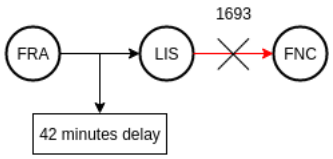


Figure 7.2: Scenario 2

The flight the passenger had missed had the following details:

- Flight Number: 1693
- Delay: 42
- Arrival: 19-04-2016 23:15

This passenger has had one previous negotiation where he argued once using a professional reason. It was also provided that the latter had referred health related issues while checking-in the original flight.

The airline provided one initial alternative from the total of twelve:

- Route: Lisbon->Porto->Madeira
- Departure: 19-04-2016 22:25
- Arrival: 20-04-2016 08:25
- Journey Duration: 10h

The rest of the procedure is the same as in the previous scenario.

7.1.3 Scenario 3

In the third and last scenario the disrupted passenger was supposedly traveling from Rome to Rio de Janeiro, having as segment flights Rome->Lisbon (FCO-LIS) and Lisbon->Rio de Janeiro (LIS-GIG) but the flight FCO-LIS underwent a thirty-five minutes delay, causing the passenger to miss his transfer from Lisbon to Rio de Janeiro as in figure 7.3.

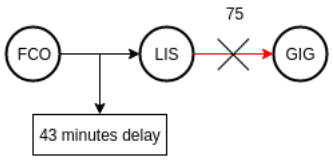


Figure 7.3: Scenario 3

The flight the passenger had missed had the following details:

- Flight Number: 75
- Delay: 35
- Arrival: 20-04-2016 08:10

This passenger has had two previous negotiation where he argued once in each negotiation using a professional reason. It is known that this passenger is a frequent traveler to Rio de Janeiro.

The airline provided one initial alternative from a total of ten:

- Route: Lisbon->Porto->Paris->Rio de Janeiro
- Departure: 19-04-2016 22:25
- Arrival: 20-04-2016 20:15
- Journey Duration: 1d1h50m

The rest of the procedure is the same as in the previous scenarios.

7.2 Results and Evaluation

To obtain realistic results ten subjects were asked to use the system in each of the previous scenarios, where information regarding each decision was recorded. In the end of the scenario testing they were also asked to answer a five question quiz in order to evaluate their satisfaction with both the solution and the use of the application. The used scenarios and questions will be presented in the appendix B.

7.2.1 Results

Despite the number of test subjects not being very high, it was possible to gather relevant information about the users and agent behaviour.

From each scenario, information was collected about each round of the negotiation and from each round, data was also gathered about the arguments used and the number of the finishing round.

Scenario 1

The results for the first round in this scenario is represented in the figure 7.4

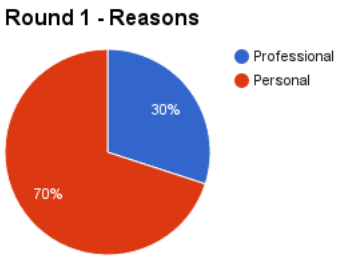


Figure 7.4: Scenario 1 - Passenger Reasons in Round 1

In the presented chart it is stated that in the first round 70% of the test subjects started an argumentation process with the PA using a personal reason, and 30% using a professional reason. In these cases the PA reacted similarly for both the personal and professional reasons. Initially the PA verified the arguments veracity as explained in 4.2. Also, for the professional reason the agent verified if any knowledge it might have would confirm that reason. If the argument was valid, it would send new alternatives matching the disrupted passenger claim. If not, a message would be sent rejecting the received argument.

In this scenario both cases are valid in the first round, so the testers moved on to the second round. The chart in figure 7.5 represents the actions taken by the testers in the second round, where it was verified that all accepted one of the newly received alternatives, finishing this way the negotiation process.

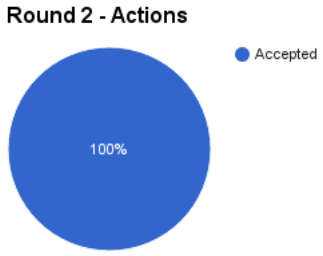


Figure 7.5: Scenario 1 - Passenger Actions in Round 2

Scenario 2

For the first round of this scenario the results are shown in figure 7.6. As it can be stated by the chart, 50% of the test population accepted some alternative in the first round. The other 50% used an health reason to try to improve the already received alternatives.

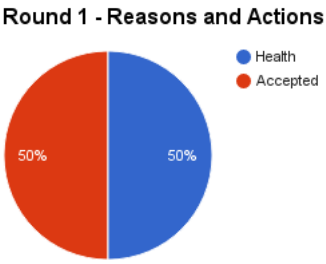


Figure 7.6: Scenario 2 - Passenger Reasons and Actions in Round 1

Since all the subjects that proceeded to the next round have used the health reason and in this scenario the passenger has specified in the check-in health related issues the argument was accepted by the agent having sent new alternatives. In the second round of this scenario and as it can be verified in the chart of figure 7.7 all the subjects that have proceeded (50%) accepted one alternative, while the other subjects have already accepted one.

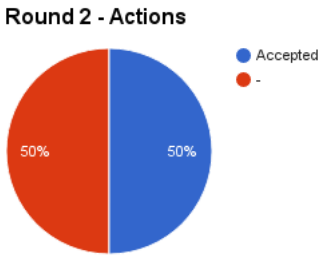


Figure 7.7: Scenario 2 - Passenger Actions in Round 2

Scenario 3

In this scenario the subjects diverged more than in the previous scenarios, allowing the agent to be tested more broadly. In the first round, the subjects were divided by the three possible reasons, and some initial acceptances as stated in the chart of figure 7.8.

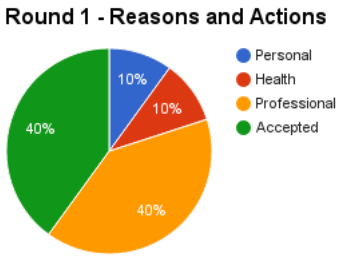


Figure 7.8: Scenario 3 - Passenger Reasons and Actions in Round 1

In this case, the subjects that used a personal and professional reason proceeded to the next round, since the arguments veracity was above the limit, and it was known that the passenger traveled frequently to that location. However the subjects that argued using an health reason saw their arguments being rejected by the agent because there was no information to back up that reason.

Despite having passed to the next round, only the subjects that did not argument using an health reason have received new alternatives, the others remained with the previous ones.

In the second round the subjects that already accepted an alternative (-) did not take action, the 10% that used an health reason argued back to the PA together with other 10% that did not accept in this round. 40% of the test subjects accepted in the second round as it can be verified in the chart of figure 7.9.

Figure 7.9: Scenario 3 - Passenger Reasons and Actions in Round 2

In the last round of this scenario and as presented by the chart in figure 7.10 the 20% of the subjects that did not accept in the previous round accepted in this one.

All the tested subjects have ended the negotiation successfully having all of them accepted an alternative with higher value than the alternative originally proposed by the company.

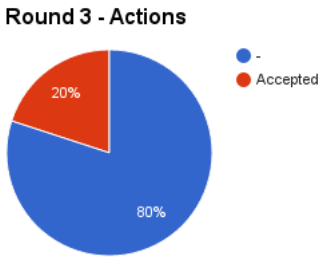


Figure 7.10: Scenario 3 - Passenger Actions in Round 3

Final Quiz

As the subjects completed the three referred scenarios, each one was asked to answer a five questions quiz on the evaluation of the system. The three first questions should be answered in a scale of 1 to 5 where 1 corresponds to Very Bad and 5 to Very Good and the question 5 with Yes/No. Those questions will be analyzed below:

1. **Compared to the original alternative how good was the new final result?**

   In this question and as can be verified in the chart in figure 7.11 the majority of the inquired claimed that the alternatives received in the end of the negotiation where classified as very good and only 20% classified them as good, being both answers above average.

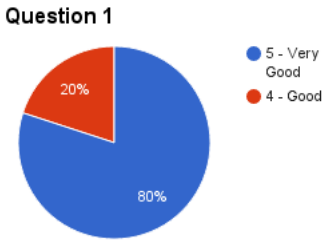


Figure 7.11: Answers to Question 1

2. **How would you classify the application for a daily basis use in case of disruption?**

   In this question 60% of the test subjects classified the application for daily basis usage as very good, where 30% classified it as good. Finally 10% of the test subjects classified the application as average.

3. **Was the argumentation process fluid?**

   When asked if the argumentation process was fluid during the negotiation 40% of the test subjects stated that it was very good where 60% claimed that it was only good as in the chart in figure 7.13.

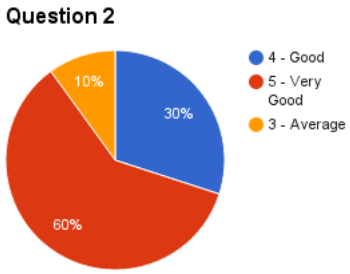


Figure 7.12: Answers to Question 2

4. **How would you improve the system?**

   The purpose of this question was to verify if the test subject had any relevant feedback regarding the system. Despite the fact that not all of the subjects have answered this question, the ones who did actually gave good answers and some of them will be referred in the next chapter regarding the future work. From the ten test subjects only five answered this question and the answers were the following:

   - Better display of each alternative
   - Broader range of arguments
   - Better interfaces
   - Record typed justification to improve accuracy
   - Taking passenger tolerance variable into account

5. **Would you recommend this system/application to others?**

   When asked if they would recommend the system/application to others all of the test subjects answered yes as can be verified in the figure 7.14.

7.2.2 Evaluation

To evaluate the work presented in this dissertation one can distinguish two parts:

- The disrupted passenger's point of view
- The airline company's point of view

From the passenger side and with the feedback obtained through the performed experiments it was possible to evaluate the system and conclude that the final satisfaction of the passenger increased greatly after negotiating and receiving new better alternatives. Besides that the classification obtained in the majority of the questions was almost always above average where a great amount of times the highest evaluation was used. Also, the fact that in all experiments the passenger accepted the solution in, at most, the third round reflects the good performance of the system.

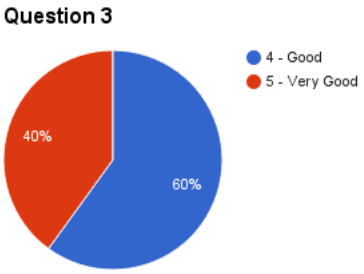


Figure 7.13: Answers to Question 3

It was not possible to receive any feedback from the airline company but as result of the used algorithm the presented alternatives were ranked based on their cost to the company. This way, even if the argument is accepted, the alternatives presented in every round are the less costly reducing as best as possible the company's losses.

7.3 Summary

In conclusion of the work presented in this dissertation this chapter presents the scenarios used to perform tests on the system. The results of this tests were also analyzed having been concluded that the subjects satisfaction highly increased after using the system by receiving new personalized alternatives after the second round.

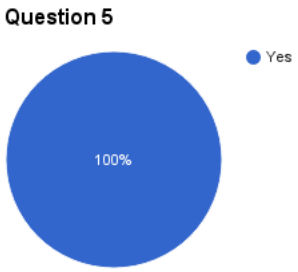


Figure 7.14: Answers to Question 5

Chapter 8

Conclusions and Future Work

This chapter will conclude this dissertation by providing a recapitulation of the contributions of the work accomplished so far and by suggesting some directions for future work.

8.1 Contributions

In chapter 1 the aim of this dissertation was described as a novel system that allows disrupted passengers to actively participate in the resolution process through the use of a mobile application and it was followed by the description of each system component in chapter 3.

Back in chapter 4 the passenger agent, entity that will act in behalf of the airline company, was defined along with its reasoning behaviours. The exchanged messages by the communicators were also defined. Still in this chapter the argumentation process was also described starting by the definition of the argument structure and followed by the presentation of the possible argument components and the detailed analysis of the algorithms used to fulfill such purpose.

The chapter 5 introduces the server that will handle the message exchange by each communicator. Since there was the need to record information from each passenger the next section of this chapter describes the database used for that purpose. It is also presented the external information used for testing purposes.

Chapter 6 introduces the mobile application that will handle all the disrupted passenger inputs in the negotiation process. Starting by describing the experience in the tested development frameworks this chapter also describes the designed interfaces for the mobile application using real examples.

Last but not least, in chapter 7 the performed experiences in the system were presented. For such case three scenarios were introduced and described. Following the scenarios the results obtained from the tests performed to ten subjects were presented along with some statistics on the answers given.

All things considered this dissertation provides contributions in two main aspects:

1. Application of argumentation in real problems using multi-agent systems.

   By using argumentation techniques in the negotiation process between the PA and the passenger an interesting view on how argumentation can be applied in real problems using a multi-agent system was provided.

2. Use of a mobile application by the disrupted passenger

   Also, the possibility for the disrupted passenger to participate in the resolution of his problem using the mobile application also points out other innovative contribution.

8.2 Future Work

Having the work presented in this dissertation as a starting point one can think of many directions for future work.

First, the argumentation process could be improved by using machine learning and natural language processing in order to open the range of possible arguments to be used, by one side to generate brand new arguments and in the other to parse any input the disrupted passenger might use.

Second, one can also think of the embedding of the developed system in the MASDIMA framework referred in chapter 2. This would be an interesting direction to go, since MASDIMA already has a proper environment on airline disruption management and besides improving the current state of MASDIMA on the passenger recovery process, it would also allow the developed system to use the information MASDIMA uses, allowing the achievement of better results.

Other suggestion for further work is to improve the current interfaces and mobile application in order for them to be production admissible. Despite being usable, the current application is rather simple and the information displayed could be improved as some of the inquired subjects referred in chapter 7.2.1.

To sum up, this dissertation has some future directions it might go for, however at the current state of the work it is already very useful for most of the airline companies in the solution of the passenger problem. The result is a full system that allows the active participation of any disrupted passenger on the resolution of his problem using a mobile device.

References

[AA] American airlines mobile application. Available at https://www.aa.com/i18n/travelInformation/traveltools/traveltools.jsp?anchorLocation=DirectURL&title=app#mobile-apps. Accessed: December 2015.

[BB06] Stephane Bratu and Cynthia Barnhart. Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling*, 9(3):279–298, jun 2006.

[BLP03] L Braubach, W Lamersdorf, and A Pokahr. Jadex: Implementing a BDI-infrastructure for JADE agents. 2003.

[BPR99] F Bellifemine, A Poggi, and G Rimassa. JADE–A FIPA-compliant agent framework. *Proceedings of PAAM*, 1999.

[CON] Codename one. Available at https://www.codenameone.com/compare.html. Accessed: December 2015.

[CRO14] António J. M. Castro, Ana Paula Rocha, and Eugénio Oliveira. *A New Approach for Disruption Management in Airline Operations Control*, volume 562 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[dC08] António Jesus Monteiro de Castro. Centros de controlo operacional : organização e ferramentas. Post-Graduation Thesis, 2008.

[Fie00] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000.

[FIP] Fipa. Available at http://www.fipa.org/. Accessed: November 2015.

[ION] Ionic framework. Available at http://ionicframework.com/. Accessed: December 2015.

[Jax] Jax-rs. Available at https://jax-rs-spec.java.net/. Accessed: January 2016.

[Jet] Jetty server. Available at http://wiki.eclipse.org/Jetty. Accessed: December 2015.

[JPNS98] NR Jennings, S Parsons, P Norriega, and C Sierra. On argumentation-based negotiation. *IWMAS98 Submission*, 1998.

[JSW98] N. R. Jennings, K. Sycara, and M. Wooldridge. A Roadmap of Agent Research and Development. *Autonomous agents and multi-agent systems*, 38(1):7–38, jan 1998.

REFERENCES

[KSE98] S Kraus, K Sycara, and A Evenchik. Reaching agreements through argumentation: a logical model and implementation. *Artificial Intelligence*, 1998.

[Mah15] Stephen J Maher. A novel passenger recovery approach for the integrated airline recovery problem. *Computers & Operations Research*, 57:123–137, may 2015.

[MAS] Masdima - multi-agent system for disruption management. Available at http://uptec.up.pt/en/company/masdima. Accessed: November 2015.

[Mon] Mongodb. Available at https://docs.mongodb.com/. Accessed: December 2015.

[TA] Turkish airlines mobile application. Available at http://www.turkishairlines.com/en-int/travel-information/turkish-airlines-mobile-applications-are-at-your-service-with-an-innovative- Accessed: December 2015.

[UA] United airlines mobile application. Available at https://play.google.com/store/apps/details?id=com.united.mobile.android&hl=en. Accessed: December 2015.

[ZH08] Yu Zhang and Mark Hansen. 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(2052):90–99, dec 2008.

Appendix A

External Data

A.1 Problem Example

```
<Table event_id="1834" event_time="2016-04-19 15:35:56" event_source="SIMULATED"
    event_type="AIRCRAFT" event_cause="MAINT" resource_affected="CSTNW"
    resource_type="320" resource_cap="156" crew_res_type="NB"
    estimated_time_to_solve="206" flight_date="2016-04-19T00:00:00" flight_number="
    661" scheduled_time_of_departure="2016-04-19 17:00:00"
    scheduled_time_of_arrival="2016-04-19 20:00:00" estimated_time_of_departure="
    2016-04-19 19:01:00" estimated_time_of_arrival="2016-04-19 22:01:00" origin="
    AMS" destination="LIS" departure_delay_in_minutes="121" bus_pax="4" econ_pax="
    142" total_pax="146" scheduled_trip_time="PT3H" estimated_trip_time="PT5H1M"
    scheduled_cost_aircraft="9301.813" scheduled_cost_crew="2002.500"
    scheduled_cost_passenger="0.000" processed="3" />
```

A.2 Problem Violation Example

```
  <Table event_id="1839" resource_affected="38ZMXJ" violation_type="1.2"
      violation_category="MISSED-CONNECTIONS" resource_type="PASSENGER"
      flight_number="75" scheduled_time_of_departure="2016-04-19 22:10:00"
      estimated_time_of_departure="1970-01-01 00:00:00" scheduled_time_of_arrival="
      2016-04-20 08:10:00" estimated_time_of_arrival="1970-01-01 00:00:00" origin="
      LIS" destination="GIG" violation_time="43" number_pax_pnr="1" />
```

A.3 Passenger Example

```
<ns3:PassengerInfo>
  <ns3:RPH>122</ns3:RPH>
  <ns3:paxRPH/>
  <ns3:paxFltSegRPH>2</ns3:paxFltSegRPH>
  <ns3:paxConSegmRPH/>
  <ns3:jumpSeatAuthority/>
  <ns3:groupRef/>
  <ns3:addToSTBYDateTime/>
  <ns3:infIndicator>false</ns3:infIndicator>
  <ns3:emplSnrtyDate/>
  <ns3:emplLengthService/>
  <ns3:accepted>false</ns3:accepted>
  <ns3:connAccepted>false</ns3:connAccepted>
  <ns3:namePrefix>MR</ns3:namePrefix>
  <ns3:givenName>Frank</ns3:givenName>
  <ns3:middleName/>
  <ns3:surname>Wolfram</ns3:surname>
  <ns3:paxTypeCode>ADT</ns3:paxTypeCode>
  <ns3:gender>Male</ns3:gender>
  <ns3:age>0</ns3:age>
  <ns3:bookStatusCode>HK</ns3:bookStatusCode>
  <ns3:bookStatusName/>
  <ns3:rebookDesigCode/>
  <ns3:bookRefType>14</ns3:bookRefType>
  <ns3:bookPNR>8H9LX6</ns3:bookPNR>
  <ns3:bookPNRContext/>
  <ns3:boardPriority/>
  <ns3:boardZone/>
  <ns3:seatCharact/>
  <ns3:seatNumber>027F</ns3:seatNumber>
  <ns3:seatClass>Economy</ns3:seatClass>
  <ns3:nonRevCateg>NRPS</ns3:nonRevCateg>
  <ns3:osi/>
  <ns3:ssr>
  <ns3:SpecialServiceRequest>
  <ns3:RPH>122</ns3:RPH>
  <ns3:SSRCode>BRND</ns3:SSRCode>
  <ns3:SSRServQuant>0</ns3:SSRServQuant>
  <ns3:SSRStatus>39</ns3:SSRStatus>
  <ns3:SSRNumber>0</ns3:SSRNumber>
  <ns3:SSRAirlineShortName/>
  <ns3:SSRAirlineTravelSect/>
  <ns3:SSRAirlineCode/>
  <ns3:SSRAirlineCodeContext/>
  <ns3:SSRAirlineDivision/>
  <ns3:SSRAirlineDepartment/>
```

```
47      <ns3:SSRText>HEALTH</ns3:SSRText>
48      </ns3:SpecialServiceRequest>
49      </ns3:ssr>
50      <ns3:tiket>
51      <ns3:TicketingInfo>
52      <ns3:RPH>122</ns3:RPH>
53      <ns3:tiktAirlineAccCode>047</ns3:tiktAirlineAccCode>
54      <ns3:tiktSerialNbr>1642469418</ns3:tiktSerialNbr>
55      <ns3:tiktType>eTicket</ns3:tiktType>
56      <ns3:tiktBookNumber/>
57      <ns3:tiktACNType/>
58      <ns3:tiktACNId/>
59      <ns3:tiktACNIdContext/>
60      </ns3:TicketingInfo>
61      </ns3:tiket>
62      <UUID>3582080</UUID>
63    </ns3:PassengerInfo>
```

A.4 Alternative Example

```
1  <ns3:AlternativeItinerary>
2            <ns3:iTineraryRPH>21</ns3:iTineraryRPH>
3            <ns3:depDate>2016-04-20T06:45:00</ns3:depDate>
4            <ns3:arrDate>2016-04-20T15:00:00</ns3:arrDate>
5            <ns3:journeyDuration>PT8H15M</ns3:journeyDuration>
6            <ns3:flightSegInfo>
7              <ns3:FlightSegAlternativeInfo>
8                <ns3:fltRPH>21</ns3:fltRPH>
9                <ns3:fltSegOrder>24</ns3:fltSegOrder>
10               <ns3:fltNumber>1050</ns3:fltNumber>
11               <ns3:depdateTime>2016-04-20T06:45:00</ns3:depdateTime>
12               <ns3:arrDateTime>2016-04-20T09:35:00</ns3:arrDateTime>
13               <ns3:numberStops>0</ns3:numberStops>
14               <ns3:smoking>false</ns3:smoking>
15               <ns3:ticketType>eTicket</ns3:ticketType>
16               <ns3:depAirpCodeContext>IATA</ns3:depAirpCodeContext>
17               <ns3:depAirpLocationCode>LIS</ns3:depAirpLocationCode>
18               <ns3:depAirpTerminal>1</ns3:depAirpTerminal>
19               <ns3:arrAirpCodeContext>IATA</ns3:arrAirpCodeContext>
20               <ns3:arrAirpLocationCode>BCN</ns3:arrAirpLocationCode>
21               <ns3:arrAirpTerminal>1</ns3:arrAirpTerminal>
22               <ns3:aircType>320</ns3:aircType>
23               <ns3:aircChange>false</ns3:aircChange>
24               <ns3:markAirlineCode>TP</ns3:markAirlineCode>
25               <ns3:markCabin>
26                 <ns3:CabinClassBookInfo>
```

```
                  <ns3:fltSegOrder>24</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Business</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>2</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>C</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>24</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Business</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>2</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>D</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>24</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>Y</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>24</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>B</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>24</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>M</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>24</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>S</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>24</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>H</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
```

```
                  <ns3:fltSegOrder>24</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>Q</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>6</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>24</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>P</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>6</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>24</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>V</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>4</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>24</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>W</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
              </ns3:markCabin>
            </ns3:FlightSegAlternativeInfo>
            <ns3:FlightSegAlternativeInfo>
              <ns3:fltRPH>21</ns3:fltRPH>
              <ns3:fltSegOrder>25</ns3:fltSegOrder>
              <ns3:fltNumber>2057</ns3:fltNumber>
              <ns3:depdateTime>2016-04-20T10:57:00</ns3:depdateTime>
              <ns3:arrDateTime>2016-04-20T13:35:00</ns3:arrDateTime>
              <ns3:numberStops>0</ns3:numberStops>
              <ns3:smoking>false</ns3:smoking>
              <ns3:ticketType>eTicket</ns3:ticketType>
              <ns3:depAirpCodeContext>IATA</ns3:depAirpCodeContext>
              <ns3:depAirpLocationCode>BCN</ns3:depAirpLocationCode>
              <ns3:depAirpTerminal>1</ns3:depAirpTerminal>
              <ns3:arrAirpCodeContext>IATA</ns3:arrAirpCodeContext>
              <ns3:arrAirpLocationCode>MAD</ns3:arrAirpLocationCode>
              <ns3:arrAirpTerminal>2</ns3:arrAirpTerminal>
              <ns3:aircType>332</ns3:aircType>
              <ns3:aircChange>false</ns3:aircChange>
              <ns3:markAirlineCode>UX</ns3:markAirlineCode>
              <ns3:markCabin>
                <ns3:CabinClassBookInfo>
```

```
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Business</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>2</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>J</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Business</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>2</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>C</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Business</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>2</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>D</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>8</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Business</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>2</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>I</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>4</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>Y</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>B</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>M</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
```

```
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>L</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>E</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>K</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>V</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>H</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>S</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>R</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
```

```
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>U</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>T</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>Q</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>9</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>25</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>P</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>5</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
              </ns3:markCabin>
            </ns3:FlightSegAlternativeInfo>
            <ns3:FlightSegAlternativeInfo>
              <ns3:fltRPH>21</ns3:fltRPH>
              <ns3:fltSegOrder>26</ns3:fltSegOrder>
              <ns3:fltNumber>1143</ns3:fltNumber>
              <ns3:depdateTime>2016-04-20T14:53:00</ns3:depdateTime>
              <ns3:arrDateTime>2016-04-20T15:00:00</ns3:arrDateTime>
              <ns3:numberStops>0</ns3:numberStops>
              <ns3:smoking>false</ns3:smoking>
              <ns3:ticketType>eTicket</ns3:ticketType>
              <ns3:depAirpCodeContext>IATA</ns3:depAirpCodeContext>
              <ns3:depAirpLocationCode>MAD</ns3:depAirpLocationCode>
              <ns3:depAirpTerminal>2</ns3:depAirpTerminal>
              <ns3:arrAirpCodeContext>IATA</ns3:arrAirpCodeContext>
              <ns3:arrAirpLocationCode>OPO</ns3:arrAirpLocationCode>
              <ns3:aircType>ER4</ns3:aircType>
              <ns3:aircChange>false</ns3:aircChange>
              <ns3:markAirlineCode>UX</ns3:markAirlineCode>
              <ns3:markCabin>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>26</ns3:fltSegOrder>
```

```
                    <ns3:cabinTypeName>Business</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>2</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>J</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>5</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Business</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>2</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>C</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>4</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Business</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>2</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>D</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Business</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>2</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>I</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>2</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>Y</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>B</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>M</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
```

```
                    <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>L</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>E</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>K</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>V</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>H</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>S</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
                    <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                    <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                    <ns3:resBookDesigCode>R</ns3:resBookDesigCode>
                    <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                  </ns3:CabinClassBookInfo>
                  <ns3:CabinClassBookInfo>
                    <ns3:fltSegOrder>26</ns3:fltSegOrder>
```

```
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>U</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>26</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>T</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>26</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>Q</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
                <ns3:CabinClassBookInfo>
                  <ns3:fltSegOrder>26</ns3:fltSegOrder>
                  <ns3:cabinTypeName>Economy</ns3:cabinTypeName>
                  <ns3:cabinTypeRPH>3</ns3:cabinTypeRPH>
                  <ns3:resBookDesigCode>P</ns3:resBookDesigCode>
                  <ns3:resBookDesigQuant>3</ns3:resBookDesigQuant>
                </ns3:CabinClassBookInfo>
              </ns3:markCabin>
            </ns3:FlightSegAlternativeInfo>
          </ns3:flightSegInfo>
        </ns3:AlternativeItinerary>
```

Appendix B

Scenarios

B.1 Scenario 1

Full Flight: AMS-LIS-OPO

Pax Name: Jorge

Pax History:

No health related issues referred

Nor professional

2 previous negotiation:

- Both with 2 arguments providing the same reason (PERSONAL)

Disruption: LIS-OPO

Flight Number: 1958

Delay: 80

Arrival: 19-04-2016 22:20

Previous Relevant Information: On previous negotiation you argumented with the same Reason (Personal)

Imagine that you're travelling to Oporto from Amsterdam and the flight AMS-LIS is delayed by 80 minutes, meaning that you lost your flight from LIS to OPO.

The airline you're travelling presents you with the following alternative with no cost:

LIS - BCN - MAD - OPO

Departure: 20-04-2016 - 06:45

Arrival: 20-04-2016 - 15:00

Journey Duration: 8h15m

By using the provided application, interact with the company in order to find a better alternative that suits your needs, if available.

B.2 Scenario 2

Full Flight: FRA-LIS-FNC

Pax Name: Frank

Pax History:

1 previous negotiation:

- Argument used professional

Has health related issues and has referred to it in the check-in process.

Disruption: LIS-FNC

Flight Number: 1693

Delay: 42

Arrival: 19-04-2016 23:15

Previous Relevant Information: Health related issues referred in the check-in.

Now imagine that you're travelling to Madeira from Frankfurt and the flight FRA-LIS is delayed by 42 minutes, meaning that you lost your flight from LIS to FNC.

The airline you're travelling presents you with the following alternative with no cost:

LIS - OPO - FNC

Departure: 19-04-2016 - 22:25

Arrival: 20-04-2016 - 8:25

Journey Duration: 10h

By using the provided application, interact with the company in order to find a better alternative that suits your needs, if available.

B.3 Scenario 3

Full Flight: FCO-LIS-GIG

Pax Name: Carminda

Pax History:

No health related issues pointed

Frequent traveller to Rio due to business motives, and reported it check in.

2 previous negotiation:

- Argument used in both professional

Disruption: LIS-GIG

Flight Number: 75

Delay: 35

Arrival: 20-04-2016 08:10

———————————————————————————————————-

Previous Relevant Information: Frequent traveller to Rio due to business motives, and reported it check in

Now imagine that you're travelling to Rio de Janeiro from Rome and the flight FCO-LIS is delayed by 35 minutes, meaning that you lost your flight from LIS to GIG.

The airline you're travelling presents you with the following alternative with no cost:

LIS - OPO - CDG - GIG

Departure: 19-04-2016 - 22:25

Arrival: 20-04-2016 - 20:15

Journey Duration: 1d1h50m

By using the provided application, interact with the company in order to find a better alternative that suits your needs, if available.

B.4 Quiz

Questions:

1. Compared to the original alternative how good was the new final result?
2. How would you classify the application for a daily basis in case of disruption?
3. Was the argumentation process fluid?
4. How would you improve the system?
5. Would you recommend this system/application to others?

Scale: 1 - Very Bad 2 - Bad 3 - Average 4 - Good 5 - Very Good

In questions 1, 2 and 3 answer using the provided scale, and in question 5 with yes/no.