FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Arquitectura de Agente de Interface com o Portal DOV

Luís Ângelo de Sá Barbosa

VERSÃO FINAL

Dissertação
Mestrado Integrado em Engenharia Informática e Computação

Orientador: Prof. Doutor Luís Paulo Reis

Co-Orientador: Mestre António Castro

Julho de 2008

Arquitectura de Agente de Interface com o Portal DOV

Luís Ângelo de Sá Barbosa

Dissertação
Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Rui Camacho da Silva (Prof. Auxiliar)

Arguente: Luís Nunes (Prof. Auxiliar)

Vogal: Luís Paulo Reis (Prof. Auxiliar)

16 de Julho de 2008

Resumo

Esta tese teve por base o estudo e concepção de um agente de interface com o Portal DOV no ano lectivo de 2007/2008 no âmbito do Mestrado Integrado de Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto.

O Portal DOV é a plataforma on-line existente na TAP de apoio e coordenação das actividades desempenhadas pelos tripulantes. Este sistema de informação é caracterizado pelo constante fluxo de informação proveniente de diversas fontes. Como fontes de informação podemos encontrar o *Roster*, ou seja planeamento das actividades dos tripulantes durante um período, até à documentação emitida internamente pelos vários órgãos e departamentos. O Portal DOV tem como objectivo proporcionar um maior nível de independência e operacionalidade aos tripulantes através da integração e disponibilização de conteúdos e serviços previamente acedidos localmente nas instalações da TAP ou através do telefone. O Agente de Interface com o Portal DOV será o passo seguinte para um maior grau de independência no que toca ao acesso atempado de informação prioritária.

O Agente de Interface do Portal DOV assume a necessidade que um utilizador do Portal DOV tem em se manter actualizado em relação ao estado actual da informação considerada importante para a execução das suas tarefas. Através do Agente de Interface do Portal DOV, o tripulante pode definir tarefas de vigia sobre alterações no estado de informação. A constituição da tarefa baseia – se em três preferências fundamentais: o tipo de conteúdo na fonte de informação, o período de actuação e a importância da tarefa. A conjugação destes conceitos transmite a opinião do tripulante no momento sobre o conteúdo.

Mais que um simples sistema de notificação, onde o utilizador simplesmente submete alertas e é notificado de alterações ocorridas, o agente tem como grande objectivo complementar a submissão de alertas através de sugestões para novas tarefas.

A construção das tarefas tem por base a captura das preferências e decisões que o tripulante toma na descriminação de conteúdos para a criação de alertas. Esta informação é registada numa base de dados partilhada e actualizada por todos agentes em que figura o perfil dos tripulantes sobre diferentes perspectivas. As perspectivas dos tripulantes são discernidas actualmente pela sua categoria profissional, que é o conjunto de atributos transparente aos utilizadores do Portal DOV abordados. Assim o agente tem a possibilidade de fazer sugestões mais indicadas aos utilizadores consoante os diferentes atributos da sua profissão.

O estudo e concepção do agente teve duas fases distintas. A primeira fase marcou-se pelo estabelecimento da arquitectura do agente através da aplicação da metodologia Gaia com os ajustamentos propostos por António Castro, a aplicação da metodologia TROPOS para definir os pré-requisitos e uma aplicação mais activa do UML 2.0 com uma notação apropriada para a construção dos modelos.

O desenvolvimento do agente teve por base as linhas definidas anteriormente e teve como ambiente de desenvolvimento o Visual Studio 2008 com recurso ao C#. O desenvolvimento teve dois aspectos fundamentais: a estrutura representativa dos dados e o processo de construção de sugestões.

Todo o processo de desenvolvimento teve como ponto de partida a definição do modelo de dados para exprimir a informação proveniente do utilizador na instituição de tarefas e processada pelo agente para a geração de sugestões.

O modelo de dados além de ser capaz de representar as tarefas de monitorização sobre as fontes de informação, deve ser capaz de exprimir a importância que cada uma das fontes de informação e respectivos elementos têm para cada perfil de utilizador.

O conceito de importância de cada elemento de uma fonte de informação baseia-se em dois conceitos, a relevância, isto é a relação directa de importância de um elemento para uma categoria, e a proximidade, ou seja a força de associação entre pares de elementos da mesma fonte de informação por uma categoria profissional. A conjugação de relevância e proximidade é o meio para elaboração de sugestões para um tripulante.

Sempre que uma tarefa é submetida, o valor da relevância e proximidade, derivados da categoria profissional do tripulante, para os elementos afectados deve ser actualizada de acordo com a prioridade da tarefa. Uma vez que foi decidido manter apenas um valor sobre a relevância actual do elemento para cada categoria, a relevância teria de absorver todos os valore submetidos até então e receber futuros novos valores submetidos.

Assim a relevância do elemento numa categoria evolui de acordo com os valores que vão sendo inseridos pelos utilizadores pertencentes à categoria. O mecanismo encontrado para actualizar a relevância baseia-se na média móvel exponencial e tem como aspecto fundamental o factor de *smoothing*, que determina a importância do valor submetido pelo utilizador no cálculo da nova relevância na categoria.

A proximidade é actualizada de acordo com o tipo de acções que o tripulante tem com o par de elementos, ou seja, a proximidade fortalece-se sempre que uma tarefa envolvendo o par de elementos é criada e enfraquece quando uma tarefa envolvendo esse par é eliminada pelo utilizador.

O desenvolvimento seguiu a linhas de orientação extraídos da fase anterior, o que permitiu atingir bastante facilidade e eficiência na concepção. A ferramenta utilizada de desenvolvimento revelou-se bastante satisfatória com as suas funcionalidades inerentes. Antes da implementação do método de geração de sugestões procedeu-se a uma simulação através de uma amostragem aleatória para localizar um factor de *smoothing* apropriado.

Apesar da concepção do agente proporcionar maior autonomia e liberdade ao utilizador, o seu valor real está directamente associado à utilização atribuída pelo utilizador e se ele realmente dá valor às sugestões encontradas.

Para tal, a curto prazo o primeiro passo será avaliar com detalhe a receptividade e satisfação dos utilizadores às sugestões devolvidas pelo agente, e gradualmente ajustar o *smoothing factor* para o padrão de comportamento dos utilizadores.

A longo prazo as orientações serão expandir as categorias de utilizadores e oferecer sugestões com recurso a perfis de utilizador cada vez mais adequados desde que se pondere o factor de escabilidade determinado durante a concepção do modelo de dados. Alternativamente, caso o método actual de geração de sugestões revele resultados insatisfatórios, este pode ser redesenhado para outro processo que pondere todas as tarefas de alertas submetidas até então uma vez que é mantido o registo das tarefas passadas.

Abstract

Portal DOV is TAP's on-line platform of support and coordination of the activities carried by the crew members. This information system is characterized by the constant flow of information proceeding from diverse sources. As information sources we can find the Roster, that is to say the activity plan for the crew member during a period, or the documentation emitted internally by official channels and departments. Portal DOV has the goal of providing a greater level of independence and operation to the crew members through the integration and enabling of contents and services previously accessed on site at the TAP installations or through the telephone. Portal DOV Interface Agent will be the following step for a higher degree of independence in concern to the timely access of priority information.

Portal DOV Interface Agent grasps the need that an Portal DOV user has in to be up-to-date about the current state of the information deemed important for the execution of his tasks. Through the Portal DOV Interface Agent, the crew member can define alert tasks about changes in the information state. The constitution of the task stands on three basic preferences: the type of content present in the source of information, the period of action and the importance of the task. The conjugation of these concepts transmits the crew member opinion at the moment about the content.

More than a simple system of notification where the user simply submits alerts and is notified of occurred events, the agent has as higher goal to complement the alert submission via suggestions for new tasks.

The elaboration of the suggested tasks is based on the capture of the preferences and decisions that the crew member takes in content discrimination when creating his own tasks. This information is registered in a database shared and updated by all agents. This shared database informs the crew members profile throughout different perspectives. The perspectives of the crew members are discerned currently by his professional category, which is the set of attributes transparent to the Portal DOV users in question. Thus the agent has the possibility to make suggestions specific to the users in agreement with the different attributes of his profession.

The study and conception of the agent had two distinct phases. The first phase was aimed to the foundation of the agent architecture through the application of the Gaia methodology with the adjustments considered by António Castro, i.e. the application of methodology TROPOS to define prerequisites and a more active use of UML 2.0 with the appropriate notation for the construction of the models.

The development of the agent had as guidelines the assessments previously made in the initial phase and had as development environment the Visual Studio 2008 with resource to the C#. The development had two key aspects: the data representation structure and the process for suggestions formation.

The development course had as starting point the definition of the data model needed to express the information proceeding from user task creation and later treated by the agent for the generation of suggestions.

Although the conception of the agent provides greater autonomy and freedom to the user, its real value lies directly associated with the use given by the user and if he really accepts to the suggestions found.

For such, at short-term the first step will be to evaluate with detail the user reception and satisfaction with the suggestions returned by the agent. This will allow to gradually adjust the smoothing factor in harmony with the standard user behaviour.

On the long run the aim will be to expand the user categories and so offer suggestions with access to user profiles evermore adjusted. Before any action is taken the factor of scalability determined during the conception of the data model must be pondered. Alternatively, in case of the current method of generation of suggestions discloses unsatisfactory results, he can be redesigned for another process that ponders all the tasks of alerts submitted until. This is possible because the data model keeps record of all tasks.

The data model beyond being able to represent the tasks monitoring the information sources, must be capable to state the importance that each information source and respective elements have to each user profile.

The concept of importance for each element of an information source relies on two concepts, the relevance, that is the direct relation of importance of an element to a category, and the proximity, that is the strength of association between pairs of elements of the same source of information by a professional category. The combination of relevance and proximity is the mean for the creation of suggestions for the crew member.

Whenever a task is submitted the value of the relevance and proximity, derived from the professional category of the crew member, for the related elements must be updated in accordance to the priority of the task. Since it was determined to keep only one value of the current relevance of the element for each category, the current relevance would have to absorb all submitted values until then and be ready for future new values.

Thus the element relevance in a category evolves with the values being inserted by the users belonging to the category. The mechanism found to update the relevance is based on the exponential mobile average and has as basic feature the smoothing factor, which determines the significance of the value submitted by the user to the computation of the new relevance.

The proximity is updated in accordance with the type of action that the crew member with the pair of elements, that is, the proximity is fortified whenever a task involving the pair of elements is created and weakens when a task involving this pair is eliminated by the user.

Acknowledgements

À voz dos meus pais e irmãos,
Ao apoio do Eng. António Castro,
À orientação do Prof. Luís Paulo Reis,
Às palavras do Jaime
Ao abrigo da Raquel
À mensagem da Marta
À escrita do Filipe
Aos cozinhados do Bruno
Às elucidações do João Vaz
À atenção da Prof. Henriqueta Nóvoa
À simpatia do António Mota
À insistência da Milene

Table of contents

1 Introduction..... 1
1.1 Motivation..... 1
1.2 Goals..... 2
1.3 Thesis Structure..... 2

2 Agents..... 5
2.1 A Definition..... 5
2.2 Taxonomy..... 6
2.2.1 Collaboration Agents..... 6
2.2.2 Interface Agent..... 7
2.2.3 Collaboration Learning Agents..... 7
2.3 Interface Agents..... 7
2.4 Conclusion..... 8

3 Information Systems..... 9
3.1 A Definition..... 9
3.2 Portal DOV..... 10
3.3 Conclusion..... 10

4 Methodology..... 13
4.1 The cycle of analysis and design..... 13
4.1.1 Goal-Oriented early requirements analysis..... 14
4.1.2 Analysis..... 14
4.1.3 Architectural Design..... 15
4.1.4 Detailed Design..... 16
4.2 Conclusions..... 16

5 Problem Definition..... 17
5.1 Portal DOV Alert Task..... 18
5.2 Problem Characterization..... 18
5.2.1 The agent's environment..... 19
5.2.2 The agent's actions..... 19
5.2.3 The agent's goals..... 19
5.2.4 Conclusion..... 19

6 Application..... 21
6.1 Early Requirements Analysis..... 21
6.1.1 Actor Crew member..... 22
6.1.2 Actor Portal DOV..... 23
6.2 Analysis..... 23
6.2.1 Subdivision of the system into sub-organizations..... 23
6.2.2 Definition of the Environment Model..... 24
6.2.3 Preliminary Role Model..... 27
6.2.4 Preliminary Interaction Model..... 31

6.2.5 Definition of the Organizational Rules..... 32
6.3 Architectural Design..... 33
6.3.1 Choosing the organizational structure..... 33
6.4 Personal Review..... 34
7 Development..... 35
7.1 Technologies..... 35
7.1.1 Visual C#..... 35
7.1.2 Microsoft Visual Studio 2008..... 36
7.1.3 Personal Review..... 36
7.2 Data Model..... 37
7.2.1 Introduction..... 37
7.2.2 Data Source..... 37
7.2.3 Agent Task..... 39
7.2.4 Crew member..... 42
7.2.5 Data Relation..... 43
7.3 Relational Database Creation and Normalization..... 47
7.3.1 Relational Database Concept..... 47
7.3.2 Database normalization definition..... 47
7.4 Relevance Processing..... 48
7.4.1 Personal Opinion Definition..... 49
7.4.2 General Opinion Definition..... 49
7.4.3 General Opinion creation process..... 50
7.5 Proximity Update..... 56
7.5.1 Proximity update by creation..... 56
7.5.2 Proximity update by elimination..... 56
7.5.3 Proximity update by change..... 56
7.6 Conclusions..... 57
8 Conclusions and Future Work..... 59
References..... 61

Table of Figures

Illustration 1.1: Portal DOV logo..... 1
Illustration 2.1: Agent Taxonomy adopted to describe an agent(adapted from Nwanna primary attribute dimension[Nwanna96a])..... 6
Illustration 2.2: Interaction Model between the user, agent and application (adapted from Maes “What is a software agent?”[Maes97a])..... 7
Illustration 3.1: Information Systems related concepts..... 9
Illustration 3.2: Portal DOV..... 10
Illustration 6.1: Crew member and Portal DOV Actors and Goals diagram..... 22
Illustration 6.2: Alert conceptual procedure..... 28
Illustration 6.3: Preliminary Roles Model..... 30
Illustration 6.4: Suggestion conceptual procedure..... 31
Illustration 7.1: Data Model concepts..... 37
Illustration 7.2: Data source building blocks hierarchy..... 38
Illustration 7.3: Roster relational schema..... 38
Illustration 7.4: Documentation relational schema..... 39
Illustration 7.5: Agent Task concepts..... 40
Illustration 7.6: Documentation task creation window..... 40
Illustration 7.7: Recurrence selection tab..... 41
Illustration 7.8: Task table structure..... 42
Illustration 7.9: Crew member characterization..... 42
Illustration 7.10: Crew member and Professional Category relationship..... 43
Illustration 7.11: Professional Category influence..... 45
Illustration 7.12: Data Relation by rank..... 47
Illustration 7.13: Example screen showing a group of suggestions.....50
Illustration 7.14: General Opinion simulation example..... 54
Illustration 7.15: Suggestion simulation example..... 54

Table of Tables

Table 6.1: Actors Table 23
Table 6.2: System sub-organizations..... 24
Table 6.3: Person data structure example..... 25
Table 6.4: Actions and plans related to Documentation and Roster.....26
Table 6.5: Actions and plans related to Information Reference and User Preferences.....27
Table 6.6: AlertPreferencesSet Role Schema..... 29
Table 6.7: DataSourceMonitor Role Schema..... 29
Table 6.8: InformationDifferenceView Role Schema..... 30
Table 6.9: AlertPreferencesProcessing Role Schema..... 31
Table 6.10: Preliminary Protocol updateCategoryViewOpinion.....32
Table 6.11: Preliminary Protocol askInformationDifference..... 32
Table 6.12: Liveness Rules (relations)..... 32
Table 6.13: Safety Rules (constraints)..... 32
Table 6.14: Organizational Structure..... 33
Table 6.15: Examples of clear relations..... 34
Table 7.1: Entity found in Roster..... 38
Table 7.2: Entities found in Documentation..... 39
Table 7.3: Number of tables need for each data source for proximity..... 46
Table 7.4: Mean and Mod values returned at the simulation..... 53
Table 7.5: Group of calculations used in the first study..... 53
Table 7.6: Group of calculations used in the second study..... 53
Table 7.7: Results given by the first study..... 55
Table 7.8: Results given by the second study..... 55

Glossary

CRM	Customer Relationship Management
SCM	Supply Chain Management
PLM	Product Life cycle Management
ERP	Enterprise Resource Planning
DOV	Flight Operations Department
PNT	Technical Crew
PNC	Cabin Crew
PDOV	Flight Operations Department Web Portal
UML	Unified Modelling Language

1 Introduction

1.1 Motivation

Portal DOV is currently a hub of information and services to all TAP crew members. It centralizes information and content drawn from diverse sources enabling users to have access to information and services previously accessed at the TAP installations or by telephone. For instance one crew member can view his schedule of work for the day or next week by simply logging to Portal at his home or in any place with an internet connection. Alternatively, if a service or department requires a new A330 Flight Manual to be made available for the pilots, it can resort to Portal DOV to publish it.

Illustration 1.1: Portal DOV logo

Portal DOV grants to the user a greater mobility and independence, but it still is limited. Because the information and documentation are regularly changing the user must be up to date about the significant changes. For instance, a crew member must take an active attitude to check up his roster for relevant changes, even though they may not occur. The process of returning to Portal DOV only to check up the current state of information can turn tedious or monotonous task.

This sort of tasks is the competence of interface agents, whose purpose is to facilitate the users work by doing boring or repetitive tasks, freeing the user to use the freed time more efficiently.

The Portal DOV interface agent allows the user to define tasks about the surveillance of information sources by different levels of priority. An example would be a flight attendant that defines an high-level alert task to monitor changes concerning the days off in her roster and very-low-level alert task associated with the publication of new internal communications from DOV. With the help of the agent she could be more promptly acquainted about changes in the current state of information seen as important to her.

The Portal DOV Interface Agent should also make suggestions to the crew member about the creation of new tasks. For instance a pilot from the fleet A320 could get a suggestion to create a very-high-level alert task about *"Manuais: AOPM"* because his peers were expecting the new airline operation manual to be published. By suggestions crew members could be more aware of significant changes in the information according to their profile. The profile would be based on the professional category of the crew member.

Portal DOV relying on interface agents to represent crew members could result in two major gains:

- even more mobility and independence to the users, as the need to access Portal DOV to verify the current information diminishes;
- the increase in the awareness about crucial information for each professional category by making suggestions to the crew member.

1.2 Goals

The Portal DOV interface agent should achieve two main goals:

- Allow the user to define properly tasks set to monitor changes in content relevant to the user and ensure the user is properly informed;
- Make suggestions perceptive to the user about new tasks;

Although there will be an interface agent representing each crew member and appointed to the vigilance of information deemed important by the owner, it does not require direct interaction with the existent agents. The agent will operate blind to the existence of other agents. This will be an interesting aspect because the Gaia methodology selected to establish the architecture is oriented to multi-agent environment.

The suggestions are based on the general opinion of the users with the same profile in regard to what is important to them. An example would be the case where a flight attendant on vacation gets a suggestion to create a new very-high-level alert task on Portal DOV communications. This occurred because in her absence happened major change in the access to Portal DOV and the next Portal DOV announcement would be extremely important to all crew members access it properly. Because our flight attendant was absent enjoying sunny days on the Fiji, without the agent she wouldn't have the chance to be aware. This inflated example illustrates one way in how the alert task and suggestions can help the crew-members of keeping up with what is important to the professional category of crew members at a time.

1.3 Thesis Structure

The body of the thesis is made up by four major sections.

In the first section we look into the definition and discussion of the two large concepts found during the project: interface agent and information system. Here is also provided a current state of the art about these two elements.

The second part we describe the current state methodology Gaia in regard to the suggestions made by Antonio Castro for improvements by the adoption TROPOS[Giorgini04] for early requirements elicitation and UML 2.0 for model presentation [Castro06].

The third part makes up the core of the thesis. It is a three step process where initially detailed problem definition is given, then a solution is discussed and conceived and finally it is implemented by with the tools and means selected.

The concluding part is where all the conclusions from all relevant chapters are rounded up to give a final word and are presented future directions.

2 Agents

2.1 A Definition

An agent is defined by several authors in distinct ways:

"A computational system which is long lived, has goals, sensors and effectors and decides autonomously which actions to take in the current situation to maximize progress toward its (time-varying) goals." [Maes97a]

"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators." [RussNorvig03]

"An Agent is a computational system located in specific an environment, whose environment perception is made through sensors, has decision capabilities, acts autonomously with his environment through actuators, and possesses high-level communication skills with other agents and/or human beings, so to be able to perform the task for which was conceived." [Reis03]

As one can easily see, there is no universal definition of agent, but rather a diverse collection of definitions, each one with his set of qualities. A precise description of agent has been subject to much debate throughout the years [Nwana96], and although these qualities can vary from definition to definition, they can be summed to outline a group of familiar traits common to any form of agency. In this paper we quote the definition given by Stan Franklin and Art Grasser:

"An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future." [FrankGraess96]

An intensive effort was to put forward by Luís Paulo Reis in order to compile all the vast information concerning the problematic definition of agent [Reis03]. It is highly recommend the reading of his work for a deeper understanding. Here the discussion is much more oriented to the nature of the agent in sight.

2.2 Taxonomy

As challenging as to clearly define what an agent is, is to devise an accurate and unique classification system of agents, or a typology of agents. In truth, an agent can be classified by the light of different dimensions, be them mobile or static, deliberative, reactive or hybrid, just to name a few.

In this work we use the classic approach offered by Hyacinth S. Nwana to classify autonomous agents looking the behaviour it should display in regarding three primary attributes. We consider that this kind of approach offers a clear and comprehensive view of similar agents and their traits touching the goal of this work. The primary attributes used to evaluate the agents are autonomy, cooperation and learning.

Autonomy is the ability of an agent to operate without the support of human assistance. Autonomy requires a set of goals and internal states so an agent can achieve the goal on behalf of its user. The human assistance can be important, especially when the agent lacks experience or knowledge. [RussNorvig03]

Cooperation is the ability to interact with other agents and possibly humans via some communication language [WoolJenn95]. Cooperation to achieve resolution of goals is the motivation behind multi-agents systems.

Learning can be seen as capability to extract new knowledge from experience resulting of the perception and interaction of the agent with his environment. This new knowledge can result in some form of adaptation by the agent and an increase in performance [RussNorvig03].

The combination of these primary attributes is used to derive the agents shown in figure2.1:

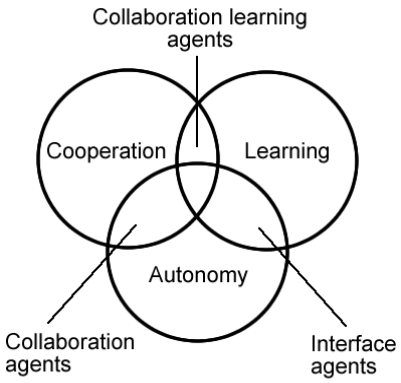


Illustration 2.1: Agent Taxonomy adopted to describe an agent(adapted from Nwanna primary attribute dimension[Nwanna96a])

The distinctions implied by the picture are not rigid ones. For instance, the focus of an Interface Agent is to be autonomous and capable to learn so it can provide the best assistance to the user, but it can exercise some cooperation to learn more efficiently.

Following we have a brief description of each category before delving into Interface Agents with more detail.

2.2.1 Collaboration Agents

Collaboration agents rely on autonomy and cooperation with other agents to achieve their goals. These agents can be applied to solve a large range of problems, where resources are limited or distributed, or the problems are too complex or spread for a single agent to handle.

The problem solving can vary from workflow management in a large company to air-traffic control.

2.2.2 Interface Agent

Interface Agent tries to support his user with computer applications interaction in order to alleviate him from repetitive or tiresome tasks or provide assistance when executing laborious tasks. It relies in learning to find out how to execute those same tasks and discover new ways of helping the user interact with the system.

2.2.3 Collaboration Learning Agents

Collaboration learning agents deal with large or inherently distributed problems where they have to work as a group but, at the same time need to be flexible to respond the changes in the environment. To offer the best response to this type of challenge both agents and their collaborative structure need to be adaptive in order to react accordingly and be able to take advantage of new opportunities that may arise in the changing environment. Collaboration learning agents can, for example, be applied in economics or game theory.

2.3 Interface Agents

Given a broader view of the agents' types through three primary attributes and with a brief explanation, the next step was to understand the organization of interface agents with a greater detail. With a more complete characterization of interface agents we hoped to obtain a clear view of the tasks implied in the resolution of this project. Maes defines Interface Agents as:

"Computer programs that employ artificial intelligence techniques in order to provide assistance to a user dealing with a particular application. ... The metaphor is that of a personal assistant who is collaborating with the user in the same work environment." [Maes94]

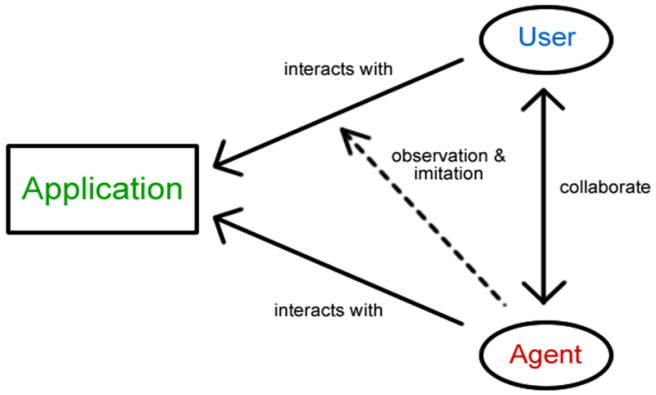


Illustration 2.2: Interaction Model between the user, agent and application (adapted from Maes "What is a software agent?"[Maes97a])

In a simpler sense, Interface Agents try to assist the users executing their tasks. In general the tasks they try to assist are repetitive, arduous or boring, that tend to generate some level of tedium or dissatisfaction.

The main features of these agents are their autonomy and learning capabilities, relying sometimes in cooperation with other agents to learn more easily or react to unexpected situations.

Maes identified several technical issues regarding the relationship between the interface agent and the user [Maes97b] being them:

- Understanding – how the user interprets the agent. This aspect is critical for the success of the collaboration between the user and the agent, as it relates to the trust between the user and the agent. To facilitate understating the agent behaviour should be simple;
- Control – how the user can define the behaviour of the agent. It should regard the autonomy of the agent and the user‘s ability to decide different characteristics of the agent so it can be more responsive to its user;
- Distraction – how the user defines the level of interaction with the agent. This question deals with the way the agent should keep the user informed and be interrupted if needed. Once again the user should be able to define the level of importance of different events;
- Ease of use – how easily the user interacts with the agent. Every interaction with the agent should be simple and friendly to the user so he can take the most of it and find its use rewarding;
- Personification – how the agent should represent the user. The agent should act on behalf of the user, and to do so, it important to grant the correct features to best represent the user personal interests.

Although Maes makes suggestions in how to deal with those issues we can see that they are intimately related, being dependent with one another and with the problem at hand. For instance, we can see how control can be applied to judge the importance of events or how the agent informs the user. By another way the agent should supply an interaction design so these definitions can be easily performed.

2.4 Conclusion

This chapter begins by giving an *incomplete*, at best, explanation of agency. Incomplete due to the multiple definitions and classifications for the concept of *“agent”*.[Reis03] The next action would be to find taxonomy to better portray the Portal DOV Interface Agent and elicit fundamental features. Explicitly Portal DOV Interface finds itself in the category of interface agent.

As expected, the Portal DOV Interface Agent will have two distinct traits. First it will allow the user to define the agent’s basic behaviour, directly related to the priority of the tasks created. Second the agent will have ability to make suggestion about the creation of new tasks, and will not interact directly with other agents.

3 Information Systems

In this chapter we offer a brief explanation to the concept of Information System. In the second part we introduce and explain the main reasons for the selection of Portal DOV, i.e. the information system behind this work.

3.1 A Definition

"Information systems are the means by which people and organisations, utilising technologies, gather, process, store, use and disseminate information." [UKAIS99]

Information Systems are systems delegated to the employment of information. While the enabling technology does have an important role, it's not crucial when the employment of information can revolve around a number of different processes, people and organizations.

Illustration 3.1: Information Systems related concepts

In the present competitive business environment having the right information and the right use for it is vital. Information Systems are more and more decisive tools which allow more efficient and effective ways of carrying out business processes and managing the business in the organizations. Information Systems have evolved through the years, always trying to adapt to either technological advances or to its mutual relations with the society [Mingers95].

Nowadays we find Information System applications toward diverse goals, each one with his unique features. CRM is oriented to the management of client and stakeholders relationships intent on the capture, storage and analysis of concerned information. SCM or PLM are especially adjusted to manufacturing companies, supervising the whole product life cycle, since his conception to disposal. Alternatively some companies like SAP AG offers ERP's, which allow integration of different systems in one global system providing the organizations with better control and awareness of their processes and people involved in the business. Information Systems can be pre made and tailored, in line to specific needs or environment.

3.2 Portal DOV

The Information System tackled in this work is one currently present in TAP Portugal, namely, the Portal DOV. We decided to choose this information system mainly by three reasons. First it offers a rich environment in constant change and evolution, where the users have real needs and this work can provide some suitable answers to them.

Secondly this information system offers the ideal test field to evaluate the project and establish final conclusions in solid ground.

Finally all the work and deriving conclusions will have direct impact and application in Portal DOV. With this, the chance of improvement and extraction of more solid conclusions in long term is more substantial.

Illustration 3.2: Portal DOV

Portal DOV, *Flight Operations Department Web Portal*, is an on-line platform of support and coordination to all the activities carried out by the crew members and the users of the services provided by the portal. Portal DOV aims to give its user a greater degree of independence and operation, giving remote access via internet to services previously accessed by telephone or locally at the TAP installations. This improved mobility enables a decrease in the effort in the task execution by the crew members and other users likewise a substantial reduction in administrative costs.

3.3 Conclusion

The importance of information systems is undeniable nowadays. [WardPepp02] They are increasingly present not only in the business but in our lives also.

Portal DOV is a growing information system providing assistance to TAP crew members performing their job. Because there is gradually more information and services as Portal DOV expands his utility, so the complexity may rise making the proper use more difficult. One way

to lessen this impact would be to provide the Portal DOV users an autonomous mechanism to filter the right information. This mechanism would have simple rules of actuation, like an alert method where the user defines content to be watched and a priority coupled.

The mechanism envisioned would confer gains in both ends making users more independent and the role of Portal DOV as an efficient information system more authentic.

4 Methodology

To establish an architecture the present work followed the steps proposed by António Castro, in *"The rationale behind the development of an Airline Operations Control Centre using Gaia based methodology"*. [Castro06]

In his paper is shown how the Gaia methodology [Zambo03] is complemented with parts of other methodologies like TROPOS [Bresciani04] and notations as UML 2.0 to provide a more complete analysis and design of a multi-agent system for an Airline Company Operations Control Centre.

4.1 The cycle of analysis and design

The whole process of analysis and design applied is comprised by a succeeding set of four steps. Each step may contain a group of processes and its deliverables, devised to support the realization of the next step. As mentioned earlier, these steps suffered some adjustments suggested by Antonio Castro to offer a better execution of the whole process like the application of the UML 2.0 or TROPOS to elicit early requirements. [Castro06]

In regard to the application of UML 2.0 Castro asserted that some previous representations could be replaced by UML 2.0 compositions, for example the formal notation representing the organisational structure or the table representation of the agent model and service model. Others are used jointly as the employment of UML 2.0 can assist to picture the organisation with their roles, activities and protocols. For example we have combined graphical representation that includes the environment model of the preliminary role diagram as a complement of the Gaia preliminary role model, or the preliminary interactions UML representation with the preliminary interaction model.

The first step taken is the establishing of requirements and their relations obtained by the application of goal-oriented early requirements of TROPOS[Bresciani03]. This stage provides a complete set of requirements necessary to the completion of the next phase, the Analysis. The Analysis comprehends the subdivision of the system into sub-organizations, the definition of the environment model, of the preliminary role model, the preliminary interaction model and finally organizational rules. Having specified the requirements is necessary to take decisions regarding the organizational structure of the multi-agent system. These choices are taken during the Architectural Design, sustained on the defined topology and control regime. In this phase the

role model and interaction model can be refined when the organizational structure is outlined. The final step in this process is the Detailed Design where is made the Agent Model and Service Model. The Agent Model shows the agents that will be implemented where the Service Model exhibits the services necessary. This specification is language/middleware neutral.

Following is presented a more complete description of the cycle of analysis and design.

4.1.1 Goal-Oriented early requirements analysis

This initial phase employs the early requirements analyses of TROPOS to get a better gathering and understanding of requirements of the system [Castro06]. In a goal-oriented requirements analysis the domain stakeholders are modelled as actors, depending on one another to achieve their goals. Plans or set of actions, creating or making use of Resources, must be carried out so the goals can be attained.

The main advantage of this approach is how it clearly identifies the several components of the system and their essential interactions. This broad and comprehensive view of the requirements and their relationships is crucial for full grasp of the Analysis phase.

4.1.2 Analysis

The analysis phase has the goal of understanding what the multi-agent system will be like. The deliverables of this phase express the functionality and operational environment of the MAS. The analysis phase consists of 5 tasks:

1. Subdivision of the system into sub-organizations – As the name implies, this task consists in the division of the system in fractions if they exhibit a behaviour specifically oriented towards the achievement of a given sub-goal or interact loosely with other portions of the system or require competences that are not needed in other parts of the system. This task depends on the deliverables provided by the early requirements analysis.

2. Definition of the Environment Model – The environment model identifies the resources and active components present in the system. Resources are components of the system that are acted upon, be it by reading, changing or extracting their values. With the resources it should be represented the actions performed to access them. This representation may have different levels of detail. Active components are components and services, like computer-based systems or humans, capable of performing complex operations with which agents have to interact. Active components should not be viewed as part of the environment but, instead, they should be *agentified*.

3. Definition of the Preliminary Role Model – The purpose of this step is to identify the functionalities and competences needed by the organization for the achievement of its goals. In this phase we must single out the preliminary roles that will be played independently of the later organization structure decided on Architectural Design. Preliminary roles can be expressed by two abstract and semi formal main classes according to their capabilities and expected behaviour. They are Permissions, or the actions allowed in the environment to accomplish the role, and Responsibilities or attributes that determine the expected behaviour of a role, divided in Liveness Properties and Safety Properties. In this phase is also important to identify any inconsistencies between the operations allowed by the environment and the ones the roles needs or must be allowed to. All these characteristics are presented by a role schema for each one of the roles identified. The preliminary role model will be finished in the Architectural Design, given place to the full Role Model.

4. Definition of the Preliminary Interaction Model – This model expresses the dependencies and relationships between the various roles in the multi-agent system organisation. This is done with one protocol definition for each type of inter role interaction. The preliminary interaction model will be also completed in the Architectural Design.

5. Definition of the Organizational Rules – Organizational rules tries to capture the general relationships between roles, between protocols and between roles and protocols. Resembling preliminary roles, organizational rules have also Liveness and Safety rules. Liveness organizational rules act like relations and define *"how the dynamics of the organisation should evolve over time"*. For example, a specific role can be played by an entity only after it has played a given previous role. Safety rules, or constraints, are *"time-independent global invariants for the organisation that must be respected"*. For example, two roles cannot be played by the same entity.

4.1.3 Architectural Design

While the Analysis supplies the functionalities and operational environment of the multi-agent system, the Design has as its task to make decisions concerning the concrete features of the system. In the foundation of these features lies the organizational structure based on the desired topology and control regime to be applied. These decisions regarding the topology and control regime will have in contemplation the specifications documents given by the Analysis phase. All the decisions made can be expressed as Gaia suggests by a coupled adoption of a formal notation and of a more intuitive graphical representation. To offer a more complete representation of the organizational structure in his work António Castro resorted to UML 2.0 having to create own mappings between UML 2.0 artefacts and abstractions or stereotypes to fully express relations.

Having the organizational structure defined the role and interaction model can be completed by examining the impact brought by the adoption of the organizational structure in the previous models. This can result in new roles interactions or changes in previous ones derived from the organisation topology and protocols emergent from the control regime defined. So both models should be reviewed by:

- Completing all activities in which a role will be involved, including its liveness and safety responsibilities;
- Defining organizational roles, whose presence was not identified during analysis and result directly from the adopted organization structure;
- Completing the definition of protocols specifying which roles the protocol will involve;
- Defining organizational protocols, whose identification derives from the organization structure.

One thing to have in mind during this process is the distinction between intrinsic and extrinsic characteristics. Intrinsic are independent of the use of the role and/or protocol in a specific organisation structure. Extrinsic are the ones who derive from the adoption of a specific organisational structure. This distinction is important in terms of reuse and design for change. This step provides a formal representation of the organizational structure including the role and interaction model and the mentioned UML 2.0 representations.

4.1.4 Detailed Design

In Detailed Design phase two final models are made, Agent Model and Service Model, showing respectively the agents that will be implemented and the services required by them. This specification is language/middleware neutral.

4.1.4.1 Agent Model

Building the agent model can be made simply by corresponding one-to-one roles and agent classes. Alternatively one can try to find a better mapping by compacting the design, be it by reducing the number of classes and instances. When doing this we must not affect the organizational efficiency, violate organisational rules or create *"bounded rationality"* problems. Although Gaia does not recommend any special notation, the Agent Model can be expressed by a simple table, specifying for each class which roles will map to it and indicating the instances of each class that will appear in the multi-agent system or by class model diagram.

4.1.4.2 Service Model

The services are draw from the protocols, activities and liveness expressions of the roles that each agent implements.

Usually, there will be one service for each parallel activity of execution that the agent has to execute.

According to Gaia, the services model requires that, for each service that may be performed by an agent, four properties must be identified: inputs and outputs, which derive from the interaction model and from the environment model, and pre-conditions and post-conditions that derive from the role safety properties as well as from organisational rules.

Inputs and outputs derive from the interaction model protocols if the service involves the elaboration of data and exchange of knowledge between agents and from the environment model if the service involves the evaluation and modification of the environment resources. Pre-conditions and post-conditions derive from the role safety and from organisational rules, and represent correspondingly restrictions on the execution and completion of the services.

4.2 Conclusions

Gaia methodology is a four step analysis process to conceive a solid multi-agent architecture. The integration of TROPOS in the early requirements is a good set off for the conception of the architecture because it enables the visualization of the problem in simpler terms. As it helps to define the principal intervening actors, their goals and the actions they must partake to achieve them, the Gaia methodology starts with guidelines more accurate. The Gaia methodology is itself a refining process were the analyst delves gradually into a more complete architecture, taking decisions in each step which will define the next phases. UML can be applied in agreement to simplify or strengthen the process while it diminishes the resultant documentation of each phase. [Castro06]

5 Problem Definition

As demonstrated there is an enormous variety of Information Systems. Each one is oriented to support a class of requirements or challenges but some do share some parallel characteristics or common types of users.

Here we select a precise user requirement which can be found in multiple information systems - the necessity to be notified promptly or accordingly on relevant content changes to their interest.

If not properly handled, this type of requirement can make the user extremely dependent on the system to perform his tasks, be them related or not. This dependency can affect the overall effectiveness of the system mostly in three ways:

- Inefficiency by the poorer use of resources, as the user could be performing other tasks;
- User processes execution stalls or slows down;
- Dissatisfaction as the user may feel is work hampered.

One solution to improve this situation would be to the information system grant an alert method.

The alert method would allow the user to define tasks in charge of monitoring relevant content changes. The process of monitoring should be able of responding suitably to the user alert requisites and the nature of content itself.

We do not delve in much detail in the description of user alert requirements and the nature of the content, because they can vary from information system to information system or to specific user alert needs. Yet we can identify some features fundamental to the alert task:

- Target – defines what (content) the task will supervise. The target can be important in the identification of other features the alert should include;
- Importance – defines the weight of the alert to the user. The importance of the task can define its behaviour or how the user is notified;
- Lifespan – sets the period of activity of the task and can be directly linked to its behaviour. For example an alert task can act only once, periodically or be always active.

With a responsive alert mechanism the user would be free to do other tasks or be more efficiently notified about the relevant content change.

5.1 Portal DOV Alert Task

In regard to Portal DOV, we selected the specific case where crew member is interested in knowing of changes in two distinct contents: his roster and released documentation. The alert mechanism should provide the means to be establish alert tasks according to the three fundamental features: target, importance and lifespan.

A task is defined by a group of fields used by the crew member to specify suitably his alert preferences. The fields available to create the alert task are:

- *Location* – allows the user to select the general content where the task will operate. The user can select between Roster and Documentation as the content source to be watched;
- *Details* – allows the user to better define the alert task. The fields shown are dependent on the location selected previously to be monitored. For each content we have the details:

1. Roster:
    - Activity – a type of activity found in the crew member roster.
2. Documentation:
    - Type of documentation – the type of document published.
    - Publisher – the service responsible for the emission of the document.

- *Start Time* and *Due Time* – these two time intervals set up the period of activity for the task. The use of these two field is optional;
- *Recurrence* – this field allows to be defined some level of task operation;
- *Priority* – the crew member applies this scale to designate a level of importance to the task.

With this basic set of requirements in mind and applying the improvements found in [Castro06] we try to identify the ideal agent architecture for Portal DOV. This agent architecture will provide the ground for the building of an Interface Agent able to supply the user with a service capable of answering his needs in terms of information change notification.

5.2 Problem Characterization

The first step taken to understand in the scope of agency was to define the three major aspects that influence the agent behaviour: environment, actions and goals. These three aspects are related to one another, as the agent's actions must support the accomplishment of its goals in the prescribed environment.

The agent will embody the user's need to acquire up-to-date information significant to the execution of his tasks. Portal DOV, the environment where the agent operates, is characterized by a regular introduction and adjustment of present information.

The information can take multiple forms like the roster, CRA messages, holidays planning or the publication of common documentation. The agent must be able to access the information visible to the user it represents, and report of any kind of changes deemed important by the user. To do so, the user must first define the agent's basic behaviour and main goals. The main goals identify the type of information significant to the user and behaviour defines the actuation of the agent to the owner and Portal DOV.

The actuation can delineate the agent's schedule of work, be it automatic, periodic or constant, or how the agent informs the user, for example by mail or instant messaging. As the

agent interacts with Portal DOV it must learn to prioritize the information presented and to expand the user's field of interest. The user is always notified of these changes before they are established.

5.2.1 The agent's environment

The agent will have to deal with a dynamic environment, where the content is updated on a regular basis. Although the agent is restricted only to the information visible to its owner, it can easily find out if there is new information in Portal DOV, making the environment fully observable and the requirement to keep track of the current state of the world nonessential.

Various agents will interact with the environment, but at this time, there will be no need for interaction between them because their actions and goals hardly require any cooperation or any kind of communication to carry them out.

5.2.2 The agent's actions

The agent must keep its owner updated, and to do so it has to do a number of tasks, like to check if there is new information in Portal DOV or to select the relevant information to the owner. The actions performed by the agent are expected to be infallible, although we can expect nefarious interference of exterior causes to affect the execution, like connection failures or database problems. Actions related to the main goals will have equal utility and cost. The agent will operate solo, not needing to communicate with other agents or even be aware of their existence.

5.2.3 The agent's goals

The agent has one main purpose: keep its owner updated with the new relevant information present in Portal DOV. While the agent has one main ambition, it will have multiple sub goals with different utilities to support the completion of its main goal. The agent is strongly committed to its goals, abandoning them only when achieved.

5.2.4 Conclusion

The architecture will rely on a discrete representation of the environment with a simple internal state of the world. Although the agent acts on a dynamic environment, it will not need to deliberate any course of action *per se*. While the user can define this feature, more important may be when to act, having the agent the ability to choose the time to search for new content, ideally when it is more expected.

The architecture will not support the interaction between agents. The actions performed by the agent are expected to be infallible and will not affect the world. By the time being, there will not be need for a utility or cost of actions.

The agent will have a maintenance goals, keeping the user informed of relevant content changes occurred in Portal DOV.

6 Application

As mentioned previously we embraced the proposals made by António Castro to establish the architecture for the interface agent [Castro06]. The proposals include like use of TROPOS in a goal-oriented early requirements analysis and the adoption of the notion UML 2.0 to complement or fully express some models. Although the Gaia methodology is particularly oriented to the conception multi-agent systems, we hope in this particular case, to offer some personal critique of the practical outcome of the application in an autonomous agent environment.

In the final of this chapter we provide an outlook about the usefulness of the proposals to the whole process of analysis and design.

6.1 Early Requirements Analysis

Following the suggestion made the TROPOS methodology was applied to elicit early requirements. The next diagram shows the requirements found by the process needed to express the relationship between the crew member and the Portal DOV concerning the need to be aware of new significant information.

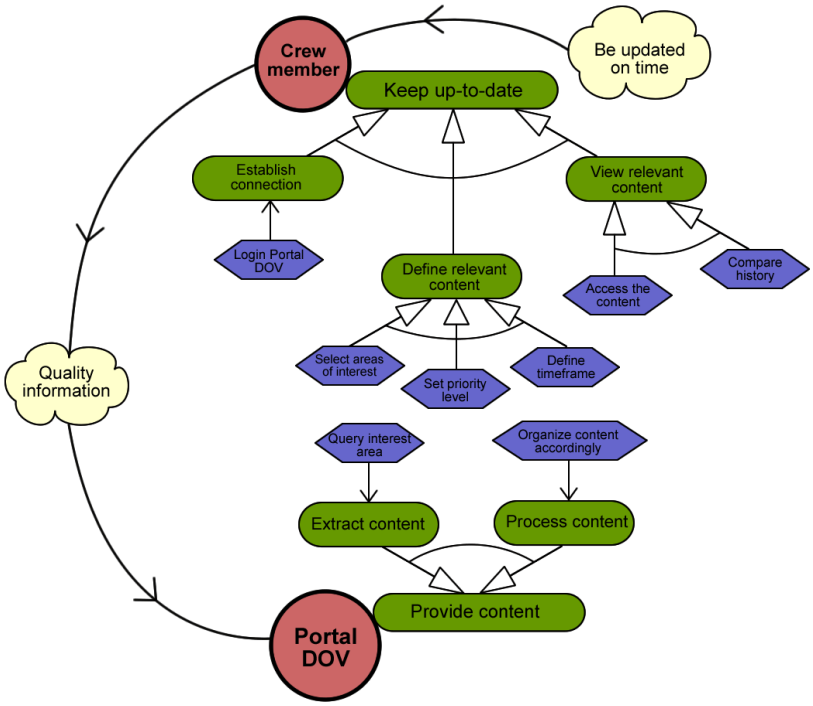


Illustration 6.1: Crew member and Portal DOV Actors and Goals diagram

In the figure 6.1 we can see the straight relationship between the crew member and Portal DOV. On one side we have the Actor *"Crew member"* who needs to access information or services to perform properly his tasks and on the other side we have Actor *"Portal DOV"* the provider of the information or services.

6.1.1 Actor Crew member

The main goal of this actor is to *"Keep up-to-date"*, meaning he must know the information needed to perform a task on time. On time is related to the soft-goal *"Be updated on time"* which can be a relative value accordingly to the importance of the information to the user himself. To be updated he must watch current state of the present information by achieving three sub-goals:

- *"Establish connection"* – first a connection with Portal DOV must be supplied so the user can have access to the content or services within the system. This is done by the plan *"Login Portal DOV"*;
- *"Define relevant content"* – to watch the status of present information, first a user implicitly must choose or decide what content present is important to him and how important it is (implicitly) in a particular timeframe. This is achieved by carrying out the generic plans *"Select areas of interest"*, *"Set priority level"* and *"Define timeframe"*;
- *"View relevant content"* – having decided the relevant content in the Information System, the next goal would be view it *per se*. Viewing the content involves the plans *"Access the content"* and *"Compare history"* of the content. The plan *"Compare history"* is applied to detect if there is any kind of significant changes to the user

The crew member success or satisfaction is strongly attached to the quality of information within the Portal DOV. This factor is linked to the soft-goal *"Quality information"*. Quality information is a relative parameter where each crew member will have different ideas when judging the value of the information concerning his tasks. Yet in our case the decisive factor when judging the quality of the information should be how timely worth is to the user, characterized by the plan *"Define timeframe"*.

6.1.2 Actor Portal DOV

Actor Portal DOV has as his main goal *"Provide content"* to its users. To the information be accessible to the users two sub-goals must be complied:

- *"Extract content"* – to provide specific content to the users the plan "Query area content" must be executed. This plan will provide the raw information concerned with the user's tasks or interests;
- *"Process content"* – before being presented to the users the content must be handled properly. To do so the plan *"Organize the content accordingly"* is executed.

From this analysis we can easily list the actors and their relations with table 6.1:

Actors	Goals	Soft-goals	Dependencies
Crew member	Keep Up-to-date	Be always updated	Information System "Quality Information" soft-goal
Portal DOV	Provide Content	Quality Information	

Table 6.1: Actors Table

In this analysis we regarded the Crew member as the active member interested in finding if there was new relevant information to him in the Portal DOV. Portal DOV was seen as having only the passive role of presenting the content. This judgment was made in order to provide more clarity in the processes the crew member takes to discern new information significant to his tasks, and consequently realize better the actions the agent acting on his behalf will have to perform.

6.2 Analysis

This chapter illustrates all the course of action taken during the analysis phase.

6.2.1 Subdivision of the system into sub-organizations

The first step taken was to clarify the relationship between Portal DOV and the crew member.

Although the early requirements analysis shed light on valuable ideas and concepts about the nature of the relationship between Portal DOV and the crew member, it also introduced contradictory notions about the role of Portal DOV and the goals of the thesis.

The application of this step was rather confusing and conflicting and some essential choices had to be made. Here are exposed some reasons.

The application of the first guideline would clearly identify two sub-organizations. One would be *"Crew member"* to achievement of the sub-goal *"Keep Up-to-Date"* and another

"Portal DOV" with the sub-goal *"Provide Content"*. While Portal DOV had the goal of providing information to the user, by the scope the study, it would be only an information repertoire with a crew member interacting periodically to verify relevant changes.

Having to face only one real *"actor"* to direct sub organizations drawing, the second and third guidelines would also prove interestingly messy. By the implicit definition of interface agent, one can easily apprehend two levels of operation, one dedicated to the interaction with the user, other to the satisfaction of the user needs. Even though they seem distinct enough, one may wonder if there are so really apart to be considered independent enough to be a sub organization. This question is especially tricky when the user interaction layer relies heavily on the layer dedicated to the user tasks to achieve their common goals, i.e. to assist the user better and better.

To deal with this question two sub-organizations were identified, the *"Crew member Interaction sub-organization"* and *"Task Satisfaction sub-organization"* to be exact. Table 6.2 concisely describes the organizations:

Sub-organization	Actor	Description
Crew member Interaction sub-organization	Crew member	This organization has primarily one goal to attain: assist the user concerning to new relevant information.
Tasks Satisfaction sub-organization	Crew member	This organization has two goals to attain: execute the tasks submitted by the user and upper layer.

Table 6.2: System sub-organizations

Once again is convenient to mention that the two sub-organizations are intimately related as they support one another actions.

6.2.2 Definition of the Environment Model

The environment model basically identifies the resources and active components present in the system. This chapter shows the tables and diagrams ensuing of the process of identification.

6.2.2.1 Active Components

There are no active components in the environment with which the agent has to interact. Although the Portal DOV acts like an active component, his role is solely of data provider and as recommended in Gaia methodology [Zambo03] was modelled in terms of resource.

6.2.2.2 Resources

Portal DOV was seen initially as an active component by the diagram *"Actors and Goals"*, but after a careful study his nature as an actor would wane. It still maintains as goal to present information or services needed by the user to complete his goals, but as the expression implies it only acts as an interface between the several data sources and the crew member. Being adjusted to the level of the main resource connected to several data sources, the next step was to explore the generic concept of Data Source.

By the light of the thesis, it was defined *Data Source* the equivalent to the pool of data and the services required to extract or process its content. The number of data sources varies from organization to organization. A data source, in the crudest view, can be seen as a grouping of data organized in a generic structure comprised in the coupling of two attributes:

- Field Name – the identification of the field;
- Data Type – the type of data stored by the field.

For example in a data source associated with one definition of a person we could have the representing structure exhibiting the coupling Field Name and Data Type as in table 6.3 :

Person		
Field Name	**Data Type**	**Description**
First Name	text	The Field Name *"First Name"* is a field storing text data.
Surname	text	The Field Name *"Surname"* is a field which stores text data.
Sex	[male/female]	The Field Name *"Sex"* is a field which stores data in the form of male or female values.
Date of Birth	{[dd-mm-yyyy]/ [yyyy-mm-dd]}	The Field Name *"Date of Birth"* is a field which stores a date in the arrangement of [dd-mm-yyyy] or [yyyy-mm-dd].

Table 6.3: Person data structure example

More complex data sources can have different hierarchies to organize their information by different levels or strategies.

This resource is associated with the fundamental feature which indicates what data source(s) the alert must check. In our case we have two data sources, Roster and Documentation, specific to Portal DOV. The data sources assessed during this phase were:

- Roster – this resource represents the roster, or schedule of activities for each crew member. This resource is periodically updated once each month although further adjustments can occur for a crew member;
- Documentation – this resource represents the library of documentation and respective services available to the crew member for access. This library is comprised of documents made known and update regularly;
- User Preferences – this resource indicates the how the fundamental features target, importance and lifespan characterize the nature of an alert. The fundamental features are distributed by the group of fields present in the definition of an alert task: location, details, start time and due time, and priority. The collection of tasks will increase as new tasks are created and it's private to the owner agent;
- Information Reference – this resource acts like a snapshot and preserves basic information about previous states of the data source associated with the alert. This reference can be used to evaluate if there is a relevant change in the present state of the information source. It should maintain a reference for each information source, the Documentation and Roster;

- Category View Opinion – this resource stores opinions based on the alert tasks details submitted by the totality of users. The Category View Opinion is organized by a group of qualities transparent to the totality of users. These qualities help to depict all the users through different perspectives and so extract suggestion accordingly. The Category View Opinion is updated whenever an alert task suffers any action (creation, change or elimination) by a crew member.

From standpoint adopted, the plan *"Query interest area"* would be salvaged from its relation with *"Portal DOV"* and replace the plan *"Access the content"* in the diagram Actors and Goals. Is important to mention services present in the Portal DOV should help the implementation of the plan.

Table 6.4 and Table 6.5 allow a more easy comprehension of the actions correlated to each plan and resource:

Resource	**Attributes**	**Action and Plan**	**Description/Conditions**
Documentation	All record	Action: READ Plan: *"Query interest area"*	When is required to find if there are relevant changes in the documentation. This condition is strongly associated with the priority of the alert task.
Roster	All record	Action: READ Plan: *"Query interest area"*	When is required to find if there are relevant changes in the roster. This condition is strongly associated with the priority of the alert task.

Table 6.4: Actions and plans related to Documentation and Roster

Resource	**Attributes**	**Action and Plan**	**Description/Conditions**
UserPreferences	AlertTaskLocation, AlertTaskDetails, AlertTaskStartTime, AlertTaskDueTime, AlertTaskPriority	Action: CREATE Plan: *"Select areas of interest"*, *"Set Priority Level"* and *"Define timeframe"*	When new alert task is placed by the crew member
UserPreferences	AlertTaskLocation, AlertTaskDetails, AlertTaskStartTime, AlertTaskDueTime, AlertTaskPriority	Action: CHANGES Plan: *"Select areas of interest"*, *"Set Priority Level"* and *"Define timeframe"*	When any alert task parameter is changed by crew member request
InformationReference	All record	Action: CHANGES Plan: *"Compare history"*	After the evaluation of the actual state the reference must be updated
InformationReference	All record	Action: READ Plan: *"Compare history"*	When is needed to find if there are relevant changes in the selected data source
UserPreferences	AlertTaskPriority	Action: READ Plan: *"Compare history"*	The priority decides how frequently this plan is executed

Table 6.5: Actions and plans related to Information Reference and User Preferences

Is important to say that at this stage the plan *"Query interest area"* has a more contextual description of his conditions when is required to evaluate if there is any change due to the multiple combinations of attributes dependent on the information source the plan deals with and the alert task definitions.

As conclusion of chapter we recognize that the active component *"Crew member"* will be the foundation for the building of the agents.

6.2.3 Preliminary Role Model

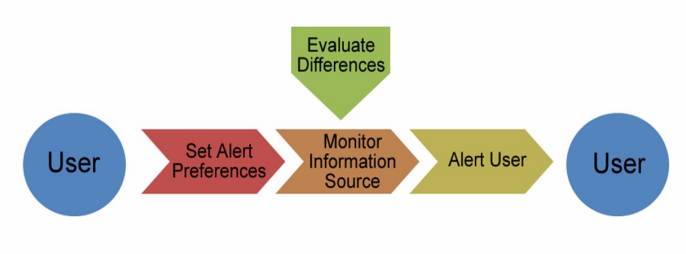
In our work we maintained the same writing terminology for the analysis and design phase in order to offer improved reading:

Roles are written in Pascal Case. Example: DataSourceMonitor;

Protocols are written in Camel Case. Example: askInformationDifference;

Activities are written in underlined Pascal Case. Example: CalcDifference.

To find out the preliminary role models more naturally, we associated them with each stage of how the Alert Procedure should proceed.



This general model exhibits one course of action accepted to guide our work in how one alert should take action. This course of action can be seen as a cycle, where initially the crew member creates the alert setting a number of preferences regarding the basic behaviour. These preferences will concern general characteristics as importance, target and lifespan. The alert will monitor periodically the data source targeted by way of the alert created by the user. The monitoring will involve a method to find if there is any change in the actual status requiring the attention of the crew member. Finally when a change in the information current state is found the user is notified in view of his settings.

By analyzing the *"Actors and Goals"* diagram and the Alert Procedure stages we straightforwardly uncover four preliminary role models:

- *AlertPreferencesSet* – this role is applied with *DataSourceMonitor* to define his basic behaviour and with *InformationDifferenceView* to judge the occurred change by the parameters defined by the crew member;
- *DataSourceMonitor* – this role has to see if there is any kind of new relevant events occurred in the chosen information source to the crew member;
- *InformationDifferenceView* – this role has to judge the difference between in the actual state of the Information Source and the one used has temporal reference. The role is used in conjunction with DataSourceMonitor. This role will operate specific algorithms and techniques according to the information source and alert settings;
- *UserAlert* – this role is applied with *DataSourceMonitor* to notify accordingly the crew member that a relevant change happened.

The tables 6.6, 6.7 and 6.8 give examples about the filling of role schemas established:

Role Schema:	*AlertPreferencesSet*
Description:	This role defines the basic behaviour of the alert mechanism by three features, the source of information it will monitor, how important is the alert to the user and the period of activity.
Protocols and Activities:	*SetDataSource*, *SetTaskPriority*, *SetTaskLifespan*, *updateCategoryViewOpinion*
Permissions:	create, update UserPreferences // to define and update alert settings
Responsibilities:	*AlertPreferencesSet = (SetDataSource.SetImportance.SetLifeSpan)^+*
Liveness:	
Safety:	data_source = new_data_source alert_importance = new_alert_importance alert_lifespan = new_alert_lifespan

Table 6.6: *AlertPreferencesSet* Role Schema

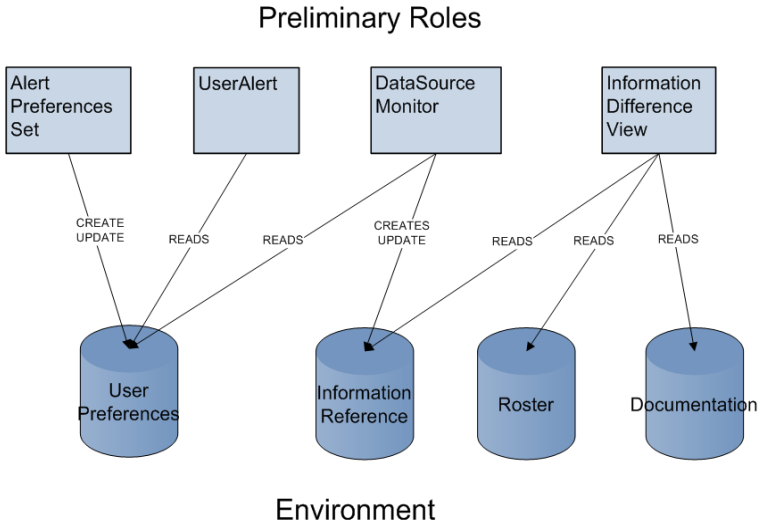
Role Schema:	*DataSourceMonitor*
Description:	This role monitors the information source for events. In case of an event verifies the difference and updates his knowledge about the current Information reference.
Protocols and Activities:	*ApplyPreferencesSettings*, *CheckDataSource*, *askInformationDifference*, *updateSourceReference*, *warnUser*
Permissions:	read UserPreferences // to behave accordingly to the users preferences read, update InformationReference // to update the current information reference
Responsibilities:	
Liveness:	*DataSourceMonitor = (ApplyPreferencesSettings.CheckDataSource .askInformationDifference)^w.warnUser|| updateSourceReference^w*
Safety:	information_reference = new_information_status

Table 6.7: *DataSourceMonitor* Role Schema

Role Schema:	*InformationDifferenceView*
Description:	This role determines if the difference between the current information state and the information reference is relevant to the user.
Protocols and Activities:	CalcDifference
Permissions:	read InformationSource // obtain the current information state read InformationReference // obtain the information reference
Responsibilities:	*InformationDifferenceView = (CalcDifference)*
Liveness:	
Safety:	

Table 6.8: *InformationDifferenceView* Role Schema

Figure 6.3 shows the preliminary role model with the roles defined above and their relations with the resources.



By envisioning how a Suggestion Procedure should act, the same strategy guided the discovery of the preliminary role models behind the extraction of suggestions of new tasks for the crew member.

The extraction of suggestions is also based in a cycle, beginning with an initial task submission with all typical preferences defined by the crew member. When the task is submitted the preferences are read, processed and catalogued by the general group the crew member belongs. The general group is a way of portraying the crew member by a perspective. The general group should be based on the crew member professional category, a trait transparent to all Portal DOV users. This means the information about the preferences of the tasks is shared between all the crew members that have identical qualities by the perspective chose to make suggestions. Finally when deemed necessary, when creating a new task for instance, suggestions about task details can be made by influence of crew members with the same professional traits.

Role Schema:	*AlertPreferencesProcessing*
Description:	This role processes the preferences of the submitted task by the crew member professional category.
Protocols and Activities:	GetCategoryViewOpinion, ProcessCategoryViewOpinion, SetCategoryViewOpinion
Permissions:	read *CategoryViewOpinion* // Get the current category opinion about the task details update *CategoryViewOpinion* // Update the category views which the crew member belongs with new values
Responsibilities:	
Liveness:	*AlertPreferencesProcessing = (GetCategoryViewOpinion.ProcessCategoryViewOpinion .SetCategoryViewOpinion)^w*
Safety:	Category_view_opinion = new_category_view_opinion

Table 6.9: *AlertPreferencesProcessing* Role Schema

6.2.4 Preliminary Interaction Model

The preliminary protocols show patterns of interactions between the roles, focusing on the essential nature and purpose of the interaction, with small consideration of execution order or message exchange order. Tables 6.10 and 6.11show the preliminary protocols and resulting UML interaction diagram derived.

Protocol name: *updateCategoryViewOpinion*		
Initiator Role: *AlertPreferencesSet*	Partner Role: *AlertPreferencesProcessing*	Input: Task preferences
Description: After the alert preferences have been set, the category corresponding to the crew member must be updated in regard to the task details by *AlterPreferencesProcessing*		Output: Yes, the update was successful OR No, update failed

Table 6.10: Preliminary Protocol *updateCategoryViewOpinion*

Protocol name: askInformationDifference		
Initiator Role: *DataSourceMonitor*	Partner Role: *InformationDifferenceView*	Input: Target data source
Description: After the alert preferences have been set, the *DataSourceMonitor* must check for relevant events in the data sources.		Output: Yes, there are is new relevant information OR No, there is no new information

Table 6.11: Preliminary Protocol *askInformationDifference*

6.2.5 Definition of the Organizational Rules

Tables 6.12 and 6.13 shown examples for set of organizational rules instituted following the guiding principles of the methodology[Zambo01].

Liveness rules	Description
updateCategoryViewOpinion (AlertPreferencesSet(taskdetails(x)) → AlertPreferencesProcessing (taskdetails(x))	Protocol *updateCategoryViewOpinion* must be executed by the role *AlertPreferencesSet* for a specific details are submitted to update the category view by *AlertPreferencesProcessing*.
AlertPreferencesProcessing (taskdetails(x)) → informCategoryViewOpinion (AlertPreferencesSuggest(taskdetails(x)))	Protocol *informCategoryViewOpinion* can only be executed by *aleterPreferencesSuggest* only before a task sharing the same details was processed by *AlertPreferencesProcessing*.

Table 6.12: Liveness Rules (relations)

Safety rules	Description
AlertPreferencesProcessing(task(x))+	The role *AlertPreferencesProcessing* must be played at least once for one task submitted

Table 6.13: Safety Rules (constraints)

6.3 Architectural Design.

6.3.1 Choosing the organizational structure

At this time the original Actors and Goals diagram would not be of use for the reasons pointed earlier. The answer is to adopt the consequent conclusions and define the main organization as a single agent for each crew member. The agent has two very close sub-organizations:

1. Crew member Interaction;
2. Tasks Satisfaction.

With the lack of active components and from the preliminary role model and organizational rules level of flexibility we it was concluded that one entity can carry all the roles. The entity can carry the roles without conflict given that obeys to the liveness and safety rules imposed.

Organization	**Topology**	**Control Regime**
Interface Agent	Centralized/flat	Autonomous

Table 6.14: Organizational Structure

At this point the application of methodology would become very puzzling. Because the roles mingle it's difficult to derive a clear relation between the roles. Although some relations seem obvious enough to identify a role like in Table 6.15, most relations seem ambiguous without having any type clearly evident. For instance, the relation *DataSourceMonitor* and *UserAlert* can be one of dependence from *DataSourceMonitor* to *UserAlert* because it relies on this role to notify the user. On the other hand it can be a relation of control because *UserAlert* only acts when deemed necessary by *DataSourceMonitor*.

The agent implicit topology and control regime also add more uncertainty compromising the further analyses. As it becomes more difficult to identify how the roles and sub organizations relate to another, the analysis of organizational structure comes to a halt. A simple but tight set of roles and relations is very hard to define as the control regime and topology center all of them on the same entity. The next decisions would come more arbitrary than a supported by foundation derived from a careful analysis.

From all the reason and feedback encountered until now the GAIA methodology, although an interesting and complete guideline to large multi-agent systems, became too much cumbersome or suffocating for simpler or autonomous agents, with few very close roles and where is difficult to detect the type of relations between the roles.

Although the Gaia methodology would not have more grounds to progress, the models and analyses made until would provide enough information to endorse a more precise conception of the agent itself. The most pertinent information would derive from the environment model were we extract the basic structure the agent will require to operate. This information will expedite the conception of the Data Model. The early-requirements analysis also provided valuable notions about the actions of the agent itself to achieve his goals. The main goals are parallel to the sub-organizations, Crew member Interaction sub-organization" and "Task Satisfaction sub-

organization” were we apprehend groups of actions oriented to a particular goal but working tightly as a one.

i, DataSourceMonitor -----depends_on--->InformationDifferenceView[i]
The DataSourceMonitor requires the knowledge from InformationDifferenceView to discern if there is a change in the information state at a moment *i* and act accordingly.
i, AlertPreferencesSugest----dependes_on--->AlertPreferencesProcessing[i]
AlertPreferencesSugest requires the knowledge from *AlertPreferencesProcessing* to have suggestions from data source *i*

Table 6.15: Examples of clear relations

6.4 Personal Review

The application of TROPOS to elicit early requirements and the methodology Gaia with the purpose of establishing agent architecture in a non-multi-agent environment had its benefits and the expected problems.

The application of TROPOS can be extremely helpful as it tries to capture key elements by two important complementary qualities, the goal and the set of actions needed by a stakeholder to accomplish it.

The problem is the characterization of the wrong relation. Although in this case the domain stakeholders are the crew member and Portal DOV, in this stage were gathered a group of expendable requirements as Portal DOV would be corrected to a different understanding as the analysis progressed. The later correction of the relation expressed earlier meant adjustment to the placement of the plans and sub-goals.

To avoid this situation my personal recommendation is to select domain stakeholders with no ambiguous nature, i.e. do not have a possibility of be seen as a resource or conflicting interpretations in future steps of the analyses.

The Gaia methodology has its advantages because allows the architecture to grow and evolve gradually through the different stages of analysis, and finally achieve very complete set models for complex multi-agent systems. Yet in rather simpler agents, either being or in non multi-agent environment with lower level of complexity, some steps can be inadequate or repetitive. The level of detail can lead to a stall as the relations between the roles the same entity implicitly will carry become harder to accurately describe. As the case is only one, before making any kind of conclusion I must maintain the observations found here as *personal review*. To reach any solid findings about the adequacy of Gaia to autonomous agents or simpler agency environments would require further study and analyses in different situations.

7 Development

The development the agent was enabled by a set of technologies, each dedicated to a layer. In the first chapter we describe briefly the technologies involved and offer some conclusion about their application. The development was made with Microsoft Visual C# 3.0 from the Microsoft .NET Framework 3.5. The editor employed was Microsoft Visual Studio 2008. The combination these tools proved to be very useful and practical with the inbuilt support to development and programming as well the easily available documentation and help on the web for more pertinent situations.

After the discussion of technologies the next chapter Data Model explains the data structure conceived to support the agent operation and interaction with the data sources derived from Portal DOV. The key to the structure is how it expresses and allows the assembly of the knowledge in the form of importance and proximity delivered by the user interaction. The Data Model is complement with a more detailed description of the relational approach adopted to create the structure.

The next step is the explanation of the mechanics behind the generation of the relevance and proximity.

7.1 Technologies

This chapter presents a brief description about the technologies used during the development phase as well a personal opinion about their benefits and weak points.

7.1.1 Visual C#

Visual C# is an object-oriented programming language, fairly modern and developed by Microsoft since 2001. Visual C# derives from the C family of languages shares similar traits with C++ and Java [CSLangSpec07]. The current version 3.0 is as a component belonging .NET Framework 3.5 and the development is mainly oriented for Windows platforms.

Although the project could be developed with a different set of technologies, like Java or a non object-oriented like C for example, Visual C# was opted due to my academic and professional background experience developing projects based on that language.

7.1.2 Microsoft Visual Studio 2008

Visual Studio is a suite of programming languages and tools for development from Microsoft. Built-in languages include Visual Basic .NET, C/C++ and C # among others. It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS to web development. As such Microsoft Visual Studio is an integrated development environment which permits programmers to create standalone windows applications, or web oriented applications and web services capable to run on any platform supported by Microsoft's NET Framework.

As Visual Studio 2008 provides a more straight integration with .NET Framework 3.5 (and as such Visual C# 3.0) the development would be much more expeditious. Although Visual Studio presents the various features and functionalities mentioned above, four would become essential during the development stage:

- Microsoft Visual Studio Debugger – A good planning can avoid many mistakes and flaws but sometimes a bug occurs which compromises the progress or affects the correct behaviour. When dealing with large chunks of code or with more complex execution logic trying to solve the error can become a serious burden. The Debugger was a critical tool because it provides very functional features to deal with this kind of situations. By allowing the coder to set breakpoints, it's given the chance to view from state to state how the execution and variables evolve, thus more clearly apprehend the erroneous behaviour. Also, letting to easily transverse the source code following the execution line, by stepping into or stepping out, makes navigation through the different functions and manual inspection during the execution fairly simple;

- IntelliSense – like other autocompletion systems, IntelliSense enables the coder to view the description of the functions and its parameters during coding by showing it in a pop-up window or tooltip. This ability saves amounts of time and makes coding less arduous because the both the need to view the complete documentation and the need to keep track of all detailed information about the coded functions is reduced;

- WinForms Designer – This project is strongly interface oriented, being imperative to have a GUI building tool to efficiently assemble a user interface capable of providing a fitting interaction. WinForms employs an event-driven programming model and provides a large range of built-in UI widgets and other kinds of controls, customizable to design a suitable interface. Supporting the visual creation and edition of the interface in design mode and granting an independent code region for devising the source code to events and specific behaviour while automatically creating the middle code layer between them confers extended separation and visibility;

- Data Designer – facilitates the graphic creation of database schemas, permitting to with ease establish tables and the relations between them. This gives a simple way to get a complete view of the data structure that supports the agent operation and change it easily if required.

7.1.3 Personal Review

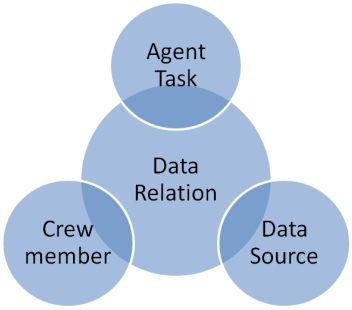
The language and IDE chosen would provide a powerful development environment with enough freedom to experiment different approaches. The present functionalities, especially the debugger, would offer valuable assistance during the development. The extensive libraries also facilitated the development greatly by proving a large diversity of methods. This was especially useful when developing the web services by the level of user transparency granted. The major fault would be quality of the available documentation about the methods. Sometimes the documentation would prove deficient or confusing having to resort to non-official channels to understand or learn the proper use of the method, or to trial-error in the worst cases.

7.2 Data Model

7.2.1 Introduction

This chapter describes the database model that supports the operation of the agent. This model is composed by four distinct concepts. Each concept is oriented to a specific goal but closely associated to the remaining.

First we have the Data Source concept, which characterizes the information the agent deals with when searching for new events. The second concept portrays the tasks submitted by the user. The third concept provides the grounds for a practical and proper crew member depiction. The fourth and final is oriented to the description of the relations between the previous concepts that compose the model and how they can enable the learning of user attitude. To each concept is shown the respective table and mentioned the most significant fields or aspects. To design the model the technique adopted was the creation of a relational database with the appliance of normalization.



In sum, the chapter focus four points: in depth information about the concepts, how they come to be, how they relate with each other and finally the database structure adopted to translate them.

7.2.2 Data Source

Data Source expresses the generic structure of the data or information derived from Portal DOV's database.

In this thesis a simplified approach to the global structure of the information present in Portal DOV was chosen due to two major factors: the scope of the thesis and the scale of the information.

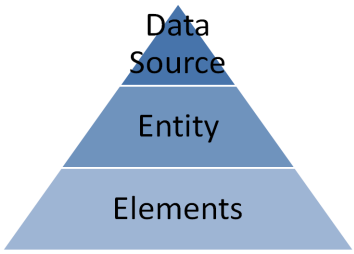
The scale and complexity of information present in Portal DOV is very large, as it has to deal with distinct kinds of information, be them flight meals and scheduled activities or documentation about the latest procedures to the Narrow Body or the A330 Flight Manual. Second, the thesis revolves around a precise kind of user, the crew member, in regard to the interaction with specific sources of information, the roster and documentation.

Even these two distinct sources of information have superfluous facts with no interest or convenience to the execution or completion of the project. Therefore there was no need to tackle the existing information in its entirety, although it could provide more accurate conclusions or different insights for a larger variety of Portal DOV users.

As such, the information present in the Data Description mimics, in a simplified way but with respectful similarity, a large fraction of the information present in Portal DOV's database. Now is provided a more detailed description of each data source.

7.2.2.1 Data Source Definition

The data sources are distinct pools of data or information, with a specific purpose and behaviour, used by Portal DOV to provide information and the services required by the crew members to execute their responsibilities. Every data source has its own entities, or abstract units that help to organize or sort out the information. The entities are themselves composed by a collection of elements.



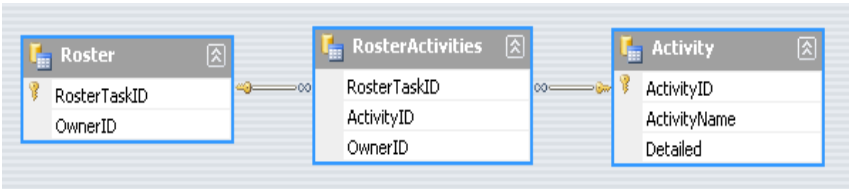
In the thesis two data sources are examined, the Roster and the Documentation

7.2.2.2 Roster

The roster comprises a set of activities a crew member will have to execute during a period of time. The crew member roster is usually planned with a 30 day advance. Sometimes, due to external causes such as bad weather or internal causes like aircraft technical problems the roster of a crew member may change. The table Activities represents the set of activities types a crew member can find in his the roster. As activities of crew member we have for example *Pairing*, *Ground Duty* or *Day Off* to name a few.

Data Source	Entity
Roster	Activity

Table 7.1: Entity found in Roster



7.2.2.3 Documentation

The documentation encompasses all the available documentation in Portal DOV. Each element of documentation is described by a pair, documentation type and publisher.

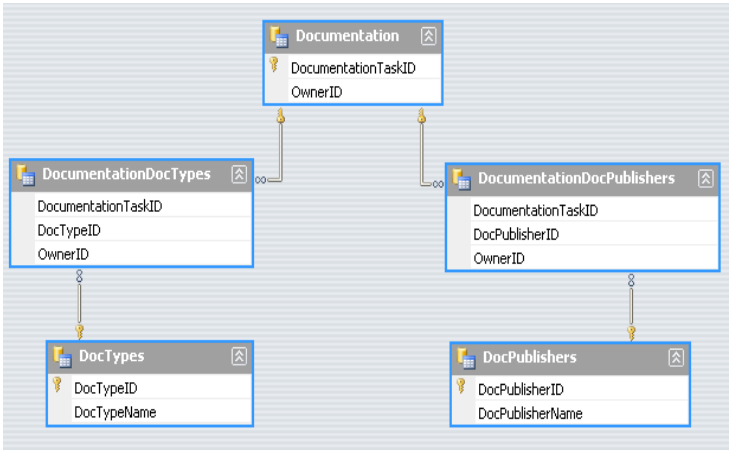
Documentation type informs the category which of document belongs to in the internal organization of TAP. It can be in the form of *Comunicação Interna* or *Planeamentos* for instance.

The publisher represents the entity or service responsible by the release of different documents found in the database. We can find *A320*, *DPC* or *Portal DOV* as publishers.

In relation to time, the documentation is periodically updated with new documents of different types submitted by the various publishers. The update can take one form of three actions: addition, change and removal of a document. Essentially, Documentation is comprised by the relation of two tables or entities: Documentation Types, table *DocTypes* and Publishers the table *DocPublishers*.

Data Source	Entity
Documentation	Document Type Publisher

Table 7.2: Entities found in Documentation



7.2.3 Agent Task

The Agent Task is the data central structure. It represents the fundamental requirement of the user, the need to be updated about events in the data sources. As mentioned earlier this need is expressed by conjugation of four concepts: who, what and where, when and how and finally importance. Now is given a more comprehensive specification to the table representing a Task and how these concepts relate to the attributes.

7.2.3.1 Who

Who is directly related to crew member, the owner of the agent, responsible for the creation of tasks for agent supervision. Crew member has its own data structures designed to support the appropriated representation. Further details can be found in chapter . In the table Task the field associated is *OwnerId*.

7.2.3.2 What and Where

What and where represents the data sources and the selection of inherent elements chosen by the user to be watched over for new events. The data sources are Roster, composed by a schedule of activities, and Documentation, the collection of types of documents submitted by various publishers.

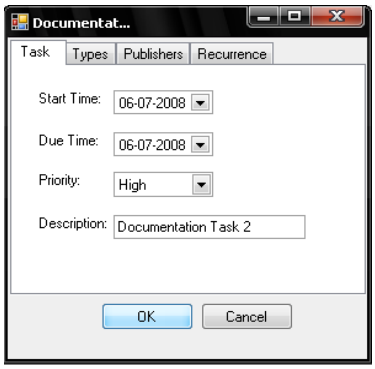


Illustration 7.6:
Documentation task creation window

7.2.3.3 When and How

When represents the task active period. The active period is usually defined by a start date, which sets the beginning of the target data source supervision by the agent, and an end date to terminate that same supervision. The period of activity is directly related with how or recurrence

type. The recurrence type defines the modus operandi of the agent in relation to time. There are four types of recurrence:

- Never – is the default recurrence type, meaning the task will perform during the time frame defined by the crew member previously;
- Number of times – the crew member can decide how many events the agent will detect of the data source chosen before expire. This recurrence type disables the activity time period;
- Until – defines a wider time frame for the task execution. The crew member needs only to define date to the task terminate. This recurrence type disables the activity time period;
- Forever – the task will run indefinitely or until the user removes it. This recurrence type disables the activity time period.

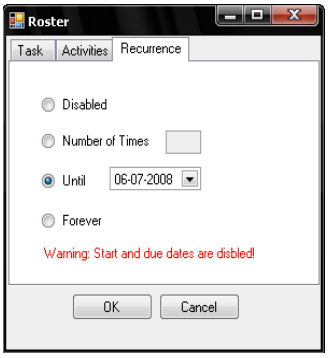


Illustration 7.7:
Recurrence selection tab

There is always one recurrence type associated to the task. The table *RecurrenceType* by the field *RecurrenceId* helps to define the recurrences.

7.2.3.4 Importance

The attribute *Importance* shows how relevant is the task to the crew member. The relevance directly influences the behaviour of the agent toward its execution and plays a major role defining how each element from an Entity relates to one another and toward the crew member. The structure used to represent the relation between how each crew member views the data sources is explained in chapter Data Relation.

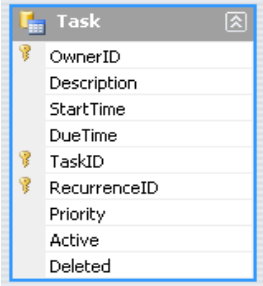


Illustration 7.8: Task
table structure

The Task itself is identified by the conjugation of the fields *OwnerId* and *TaskId*. The other fields present, like *Deleted* or *Active*, are internal for the internal operation of the task.

By the reasons given above and remembering that the Task expresses the main purpose of the agent, one can easily see that Task serves as a bridge between the different concepts present in the data structure.

7.2.4 Crew member

The concept "*Crew member*", as the name implies, represents the application user. The crew member is organized in three angles: in terms of application use, basic personal information and occupation or place within a flight.

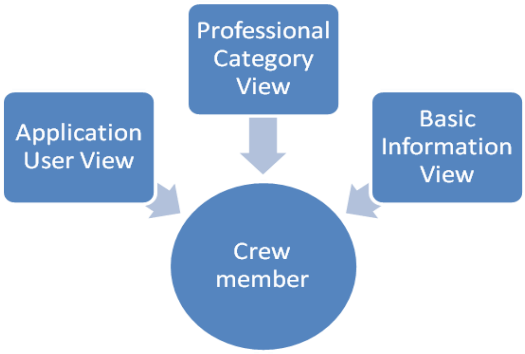


Illustration 7.9: Crew member characterization

7.2.4.1 Application User View

This view shows the information required to the user access and interact the agent. This structure is made with the usual fields of information required to depict the user in terms of use of the application, *Username* and *Password*, and one internal field of identification, the *OwnerId*.

7.2.4.2 Basic Personal Information

This view just exhibits personal information about the user like first name and last name. It does not have any practical use in terms of application.

7.2.4.3 Professional Category

Professional Category illustrates the user in terms of position in a flight. The illustration adopted bears close proximity to the real world and the flight service hierarchy found in Portal DOV. The three fields employed are: *Pool*, *Crew Group* and *Rank*.

- *Pool* – represents the type of aircraft the crew member flies. *Pool* was took as fleet in the Data Model;
- *Crew Group* – informs the type of task carried by the crew member in flight. There are two groups, *PNT* and *PNC*. PNT or *Technical Crew*, is responsible for the operation of the aircraft and the flight itself. PNT has the pilot and co-pilot ranks. PNC, *Cabin Crew*, is responsible for the comfort and well-being of the passengers, providing

assistance or service when required. Here we have supervisor, purser and flight attendant;

- *Rank* – is place in the chain of command within a crew group. The crew group has its own chain of command. For example, in PNT the pilot directs the flight and co-pilot provides the assistance. In PNC the supervisor monitors the performance of the team while the flight attendant deals directly with the passengers.

The Professional Category is given by rank and pool. Because the two groups have different ranks within, the rank can be used to identify which crew group the crew member belongs to.

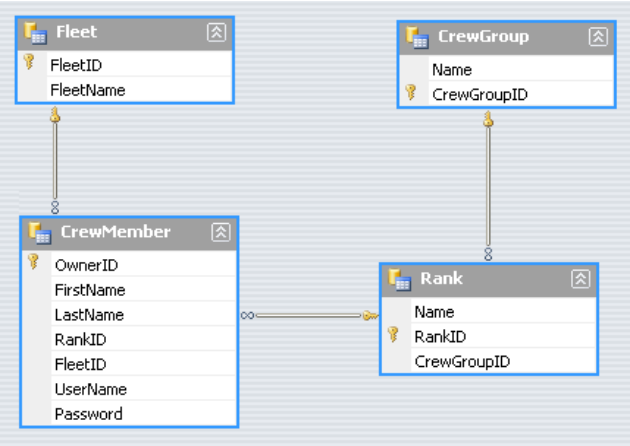


Illustration 7.10: Crew member and Professional Category relationship

This view is a fundamental aspect of the extraction of information about crew member attitude toward the data sources and constituent elements. As is explained in the next chapter Data Relation, the combination Rank and Pool makes up the prime aspect in the crew member characterization in concern to the judgment of the information present in Portal DOV and so promote adequate suggestions.

7.2.5 Data Relation

The Data Relation is the fundamental path to attain knowledge about the considerations of the crew member and the constituent professional classes in regard to the data sources when creating the tasks. The Data Relation provides the interface agent with the means to extract suggestions to the crew member about new tasks.

The knowledge is captured by the conjugation of three concepts: relevance and proximity by a view. These three concepts serve as a function to transmit the weight of a relationship between elements of a data source and a view derived from the professional category. The weight function has the generic form:

$$Weight = F(Elem(Type(k, ds), v))$$

equation 7.1: Generic Weight function

Where:

- *F* – is the function applied to extract the weight of the data source element(either the relevance or the proximity function);
- *Type(k, ds)* – function to return an entity *k* part of the data source *ds* structure;
- *Elem(Type(k, ds)* – is the function that returns a element from the data source *d*s. The variable *ds* can be one of two values: Roster or Documentation. The value returned by the function varies according to the data source chosen and how many entities make up the data source;
- *v* – the view represents the professional category chosen for interpretation the element data source. At the foundation of the professional category is the concept rank;
- *Weight* – the weight value of the relationship returned by the function *F* employed. The weight value is derived primarily from reading of the element in a data source in relation to a professional category.

7.2.5.1 Relevance

Relevance, *or how significantly the data source element is employed*, is the function revealing the importance one element present in a data source has to a professional category.

$$Relevance = R(Elem(Type(k, ds), i), v)$$

equation 7.2: Relevance function

Where:

- *Type(k, ds)* – function to return an entity *k* part of the data source ds structure;
- *Elem(Type(k, ds), i, v)* – function that returns the element *i* from the entity present *k* in data source *ds*. The variable *ds* can be one of two values: Roster or Documentation. The value returned by the function varies according to the data source chosen and how many entities make up the data source;
- *v* – view representing the professional category chosen for interpretation the element data source. At the foundation of the professional category is the rank;
- *Relevance* – value obtained by the application presenting how significant is the specific entity *i* of the data source chosen is by the professional category viewpoint *v*.

7.2.5.2 Proximity

Proximity, *or how strongly each element from the data source is associated to another when defining task*, informs how frequently each element present in a data source relates to another by the professional category standpoint. To calculate the weight from the relationship the function *Proximity* requires two different elements from the same data source.

$$Proximity = P(Elem(Type(y, ds)\ i, v), Elem(Type(z, ds), j), v), if(y = z) \rightarrow i \neq j;$$

equation 7.3: Proximity function

Where:

- *Type(k, ds)* – function to return an entity *k* part of the data source ds structure;

- *Elem(Type(y, ds), i, v)* – is function that returns the element *i* from the entity *y* present in data source *ds*. The variable *ds* can be one of two values: Roster or Documentation. The value returned by the function varies according to the data source chosen. The entities may be different given that both are present in the same data source. When addressing the same entity *(y=z)*, variables *i* and *j* must be different;
- *v* – view representing the professional category chosen for interpretation the element data source. At the foundation of the professional category is the concept rank;
- *Proximity* – value obtained by application of the function, it represents the strength of the relationship between two distinct elements from the same data source in regard to a professional category.

7.2.5.3 Data Relation Structure

To capture the relationships between the data source elements and the professional categories was needed to establish a structure capable of supporting the functions of *relevance* and *proximity* in each data source. The central component in the data relation structure would be the professional category viewpoint used to evaluate each data source element, namely *pool*, *crew group* or *rank*.

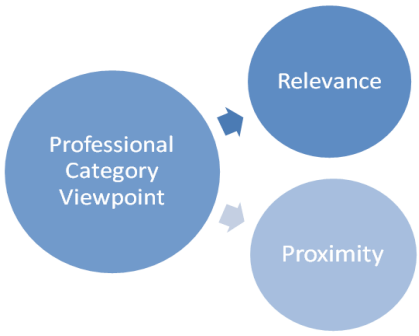


Illustration 7.11: Professional Category influence

The professional category chosen to define the viewpoint between the three was rank. The reason to the choice made was expediency, since through the rank we can also establish the relevance and proximity based on the crew group viewpoint with reduced effort.

The number of tables needed to express relevance and proximity in each data source by a viewpoint is proportional to the number of elements types present in the same data source.

For *relevance* the number of tables is given by *n*, where n is the number of entities present in the data source.

For *proximity* the tables must express the closeness in each entity and between pairs of entities. So the total number of tables follows the progression:

$$T_{tables} = n + \frac{n!}{2(n-2)!}$$

equation 7.4: Total number of tables by number of entities

, where *n* is the number of entities present in the data source.

Data Source	Entities number	Tables required
Roster	1	1
Documentation	2	3

Table 7.3: Number of tables need for each data source for proximity

In the worst case scenario the total number of tables necessary to form Data Relation structure is:

$$nViews * \sum_{i=1}^{nDatasources} 2 * nElemTypes(i) + \frac{nElemTypes(i)!}{2(nElemTypes(i)-2)!}$$

equation 7.5: Worst Case Scenario

, where:

- *nViews* – number of viewpoints to judge the entities;
- *nDatasources* – number of data sources;
- *nElemTypes(i)* – number of elements types existent in data source *i*.

For example in our case, with one viewpoint and 2 data sources, by application of the formula the total number of tables would be *1*(2+2*2+1) = 7*.

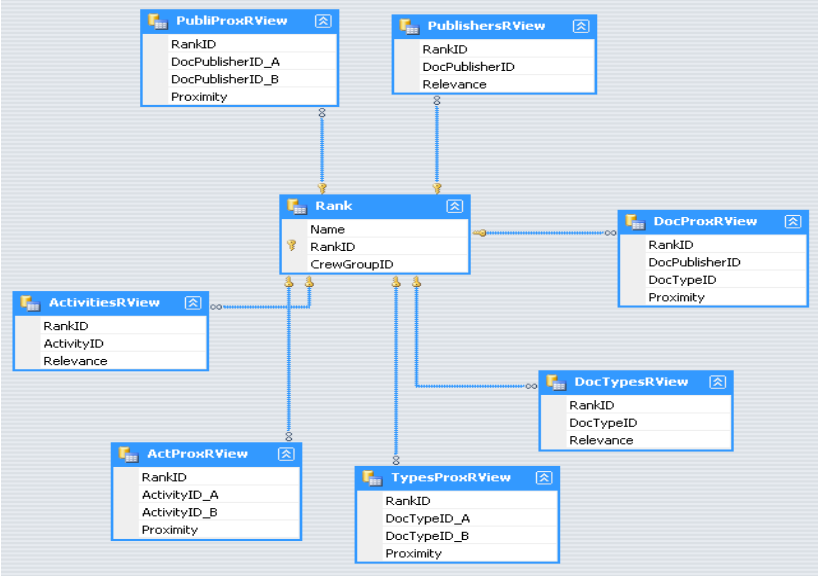


Illustration 7.12: Data Relation by rank

Due to the nature of required number of tables to express proximity is recommended to try different strategies when designing the data relation, either by creating viewpoints able to capture others viewpoints or trying to keep the number of entities in each data source at minimum.

7.3 Relational Database Creation and Normalization

This chapter explains the logic behind the relational database and gives a brief description about database normalization. Although the relational model with the normalization was adopted to express the data model they were not fully complied. Some tables have primary keys outside their true scope for easier data manipulation and access.

7.3.1 Relational Database Concept

The main purpose of a relational database is to store data in tables. Tables are organized into columns, and each column stores one type of data. Tables typically have keys, one or more columns that uniquely identify a row within the table. These tables provide a systematic way of accessing, managing, and updating data stored.[Date03b]

The most common method to manipulate relational databases is executing Structured Query Language (SQL) statements to it.

7.3.2 Database normalization definition

Database normalization is a technique to design relational database tables providing criteria for determining a table's degree of vulnerability to logical inconsistencies and anomalies. Database theory describes a table's degree of normalization in terms of normal forms of successively higher degrees of strictness. A table in third normal form (3NF), for example, is consequently in second normal form (2NF) as well, but the reverse is not necessarily the case.

During the formation of the relational database of the project were exercised the three basic rules: *1NF*, *2NF* and *3NF*.

7.3.2.1 1NF – Eliminate repeating groups

A relational database table that adheres to 1NF is one that meets a certain minimum set of criteria. These criteria are basically concerned with ensuring that the table is a faithful representation of a relation and that it is free of repeating groups.

There are five conditions to meet the 1NF [Date03]:

- There's no top-to-bottom ordering to the rows;
- There's no left-to-right ordering to the columns;
- There are no duplicate rows;
- Every row-and-column intersection contains exactly one value from the applicable domain (and nothing else);
- All columns are regular (have no hidden components).

7.3.2.2 2NF – Eliminate redundant data

A table is 2NF if it is in 1NF and all non-prime attributes are fully dependent on primary key(s) or on each candidate key. A non-prime attribute is an attribute that does not belong to any candidate key.

7.3.2.3 3NF – Eliminate columns not dependent on key

A table is 3NF if it is in 2NF and every non-prime attribute is directly dependent on every key of the table. All attributes that are not dependent upon the primary key must be eliminated.

7.4 Relevance Processing

Relevance Processing consists in revising the relevance of a data source element when a new task is submitted by a crew member. The revision is directly aligned to the view derived from the crew member professional category.

Whenever a new task is created, a relevance value is given directly by the crew member. The relevance has the goal of establishing how significant the task is to the user and so, define the agent behaviour to act accordingly. This relevance defines the importance of a set of elements of a data source at the moment to the user.

When the task is accepted at the server, the Data Relation structure, which expresses how the elements relate to the users and between themselves, must be updated to include the new relevance and so evolve as the crew members attitude about the same elements changes.

How the elements relate to each other by a professional category view is the support to elaborate suggestions to the crew member about new tasks.

For the sake of understanding lets adopt an alternative naming convention, i.e. the idea of "*personal opinion*" and "*general opinion*".

7.4.1 Personal Opinion Definition

For now, we shall take the relevance deriving from the user interaction as the crew member *"personal opinion"* at the moment in regard to a set of elements.

$$Op = R(Elem(Type(k, ds), i), ProfView(cm), Crewmember), i \rightarrow 1 \dots total$$

equation 7.6: Personal Opinion function

- *ProfView(cm)* – function which extracts the view representing the professional category from the crew member *cm* chosen for interpretation the element data source. At the foundation of the professional category is the rank;
- *Type(k, ds)* – function to return an entity *k* part of the data source *ds* structure;
- *Elem(Type(k, ds), i, ProfView(cm))* – function that returns the element *i* from the entity present *k* in data source *ds*. The variable *ds* can be one of two values: Roster or Documentation. The value returned by the function varies according to the data source chosen and how many entities make up the data source;
- *Op* – is the relevance value agreed by the crew member cm to the element *i* from the entity *k* part of the data source ds structure.

7.4.2 General Opinion Definition

The value of the relevance found in Data Relation, for the same set of element viewed by a professional category to which the crewmember belongs, we shall call *"general opinion"*.

$$Og = R(Elem(Type(k, ds), i), ProfView(cm), Data\ Relations), i \rightarrow 1 \ldots total$$

equation 7.7: General Opinion function

- *ProfView(cm)* – function which extracts the view representing the professional category from the crew member *cm* chosen for interpretation the element data source. At the foundation of the professional category is the rank;
- *Type(k, ds)* – function to return an entity *k* part of the data source *ds* structure;
- *Elem(Type(k, ds), i, ProfView(cm))* – function that returns the element *i* from the entity present *k* in data source *ds*. The variable *ds* can be one of two values: Roster or Documentation. The value returned by the function varies according to the data source chosen and how many entities make up the data source;
- *Og* – is the relevance stored in Data Relation which encompasses all the previous ones about element *i*. The value shares the same properties in relation to the *"personal opinion"*.

7.4.3 General Opinion creation process

Because it was opted to keep only one general opinion at a time, the method defined to deliver a new general opinion would be inspired by the exponential moving average. This version would follow the percent-based exponential moving average. The key issue in the process of generating the new general opinion would be the smoothing factor α applied.

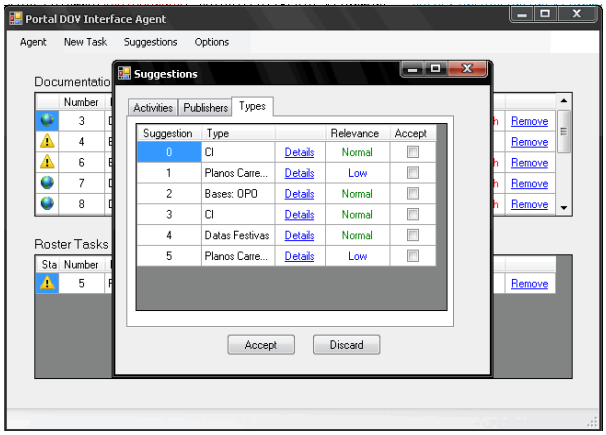


Illustration 7.13: Example screen showing a group of suggestions

This chapter provides full description about the steps taken and the conclusions obtained by the statistical analysis employed to choose an adequate smoothing factor.

7.4.3.1 The Statistical Analysis Procedure

The statistical analysis procedure adopted follows the five steps method mentioned in *Estatística* by the authors Rui Guimarães and José Sarsfield Cabral [GuimCabral97]. The analysis procedure has the subsequently phases:

1. Goal settings and conditions, where are explained the goals of the study and the environment inherent to the object of analyses;
2. Definition of data gathering process, which informs how the procedure will occur to collect the data needed;
3. Data gathering, the process itself of collecting the data;
4. Data analyses, concerning the data processing and examination at the light of the goals set initially;
5. Ascertainment of conclusions about the population, by the interpretation of the information acquire in the previous step.

7.4.3.2 Goal settings and conditions

The main purpose of this study was to select the most acceptable method by comparing their behaviour in a simulation with different smoothing factors to the average personal opinion. Another interesting condition to observe was how the very short memory term would affect the extraction of general opinions.

All the methods were subordinated to the structure imposed by the Data Relation and so only one record of the general opinion would be stored at a time. This factor although limiting, could be dealt without difficulty given the following conditions were met:

- Small classification range of the opinions, to limit possible irreconcilable deviations. In this case the range would be in a fairly manageable five qualities scale: V*ery low, Low, Normal, High, Very High*;
- General opinions included all the previous past general opinions and personal opinions feedback. This condition is mandatory and is inherent to all the methods devised;
- Personal opinions would not deviate too much when addressing opinions to the same elements. Given the relative small scale, only dealing with extreme opinions would be of concern.

The ability to react to greater and punctual deviations is the dependent in two interconnected factors:

- The structure of the data used to capture the general opinions;
- The smoothing value chose for calculation of new general opinions with new personal opinions.

While the personal opinions are based on five level strata according to their importance, all the intermediate calculations about the general opinion can work with different precisions. When needed a suggestion the general opinion obtained would be converted by rounding to a discrete value in the same five level scale.

$$Sug_i = round(Og_i)$$

equation 7.8: Suggestion function

The ability to translate later the data to a proper discrete group enables the future general opinion to be more sensitive and consistent with the evolution of the opinions given by the crew members. By converting the general opinion to a suggestion via rounding allows the five groupings have the same dimension and so ensure that bias toward a discrete value is avoided.

The exponential moving average has the following outline:

$$EMA_t = EMA_{t-1} + \alpha(Price - EMA_{t-1})$$

equation 7.9: Exponential Moving Average

, where:

- *EMAt* – new exponential moving average value;
- *EMAt-1*– previous exponential moving average value;
- Price – current value given;
- α – percent-based smoothing factor, α∃]0,1].

Equation 7.10 shows the ensuing formula which drove the calculation of new general opinions. As mentioned before, the method is heavily based on exponential moving average with a factor of smoothing α. As before, the smoothing α determines how the general opinion should shift toward the new opinions.

$$Og_i = Og_{i-1} + \alpha(Op_i - Og_{i-1}), \alpha \exists]0,1]$$

equation 7.10: New General Opinion

, where:

- *Ogi* – new general opinion value;
- *Ogi-1* – previous general value;
- *Opi-1* – submitted personal opinion;
- α – percent-based smoothing factor, α∃]0,1]

The main purpose was to directly observe the behaviour of the method with different values α=*0,5*, α=*0,33* and α=*0,66* to select the most close to the average personal opinion in a sequence of personal opinions submission.

Because only one record is stored, the general opinion must be recalculated every iteration for the new personal opinion.

7.4.3.3 Definition of data gathering process

The data gathering process would have as basis the simulation of *personal opinion* submission for the same entity. Every simulation was established to evaluate the behaviour of the methods in regard to two outcomes: how the *general opinion* and the *suggestion* made would respond after a *personal opinion* was submitted.

The data gathering process consisted in observing the outcomes *Og* and *Sug* through the application of different factor α to the submission of *n personal opinions*. The number of personal opinions defined to the simulation was 50. The recorded values would be compared by the perspective of the mean average *(X)* and mode *(Mod)*.

The random sampling was utilized to avoid bias introduction [GuimCabral97] as the typical pattern of submitting personal opinions was not known.

Each *personal opinion* had the value scale of submission translated from a quality based to a discrete scale from 1 to 5 (very low to very high accordingly). The same discrete scale was

used to convert *general opinion* to *suggestion*. The general opinion retains the value precision of 6 decimal places. The entity would have a neutral *general opinion* value in the start.

In the base simulation two descriptive statistics measurements were inspected:

- *X* – the average mean provides a rough estimate about the average opinion of the users and the behavior of the methods applied to extract general opinions and suggestions. The average has a two decimal places precision;
- *Mod* – the mode indicates the value or range of values where the data concentration is bigger. In the specific case tells about the crew members most submitted value and general opinion and suggestion value made. The mode has a one decimal place precision.

Personal Opinion	$\alpha = 0{,}5$		$\alpha = 0{,}33$		$\alpha = 0{,}66$	
$\bar{x}(Op_i)$	$\bar{x}(Og_i)$	$\bar{x}(Sug_i)$	$\bar{x}(Og_i)$	$\bar{x}(Sug_i)$	$\bar{x}(Og_i)$	$\bar{x}(Sug_i)$
$Mod(Op_i)$	$Mod(Og_i)$	$Mod(Sug_i)$	$Mod(Og_i)$	$Mod(Sug_i)$	$Mod(Og_i)$	$Mod(Sug_i)$

Table 7.4: *Mean* and *Mod* values returned at the simulation

The stored table data would be processed in three groups of study involving the pairs *(Op$_i$, Og$_i$)* and *(Op$_i$, Sug$_i$)* from each α and a process of study.

First process of study

The first study determines the global average difference between the average personal opinion and the general opinion and suggestion of the Simulation Record Table record by record.

$\bar{x}\,(\lvert\bar{x}(Op) - (Og_i, \alpha)\rvert), i\exists[1..50]$	$\bar{x}\,(\lvert\bar{x}(Op) - (Sug_i, \alpha)\rvert), i\exists[1..50]$
$\begin{cases} \bar{x}\,(\lvert\bar{x}(Op) - (Og_{i,\alpha=0,5})\rvert) \\ \bar{x}\,(\lvert\bar{x}(Op) - (Og_{i,\alpha=0,33})\rvert) \\ \bar{x}\,(\lvert\bar{x}(Op) - (Og_{i,\alpha=0,66})\rvert) \end{cases}$	$\begin{cases} \bar{x}\,(\lvert\bar{x}(Op) - (Sug_{i,\alpha=0,5})\rvert) \\ \bar{x}\,(\lvert\bar{x}(Op) - (Sug_{i,\alpha=0,33})\rvert) \\ \bar{x}\,(\lvert\bar{x}(Op) - (Sug_{i,\alpha=0,66})\rvert) \end{cases}$

Table 7.5: Group of calculations used in the first study

Second process of study

Having as reference the mean average personal opinion, the second study finds how the general opinion and suggestion mean average deviates in the Simulations Record Table.

$\lvert\bar{x}(Op) - \bar{x}(Og_{i,\alpha})\rvert, i\exists[1..50]$	$\lvert\bar{x}(Op) - \bar{x}(Sug_{i,\alpha})\rvert, i\exists[1..50]$
$\begin{cases} \lvert\bar{x}(Op) - \bar{x}(Og_{i,\alpha=0,5})\rvert \\ \lvert\bar{x}(Op) - \bar{x}(Og_{i,\alpha=0,33})\rvert \\ \lvert\bar{x}(Op) - \bar{x}(Og_{i,\alpha=0,66})\rvert \end{cases}$	$\begin{cases} \lvert\bar{x}(Op) - \bar{x}(Sug_{i,\alpha=0,5})\rvert \\ \lvert\bar{x}(Op) - \bar{x}(Sug_{i,\alpha=0,33})\rvert \\ \lvert\bar{x}(Op) - \bar{x}(Sug_{i,\alpha=0,66})\rvert \end{cases}$

Table 7.6: Group of calculations used in the second study

7.4.3.4 Data gathering

The simulations to gather random samples were prepared and executed in the spread sheet Microsoft Excel 2007. The base simulation comprised the submission of 50 personal opinions. With the data collected it would be possible to select the method with the fittest behaviour.

The next graphics feature examples of general opinions and suggestions conception by the different methods resulting from the simulation.

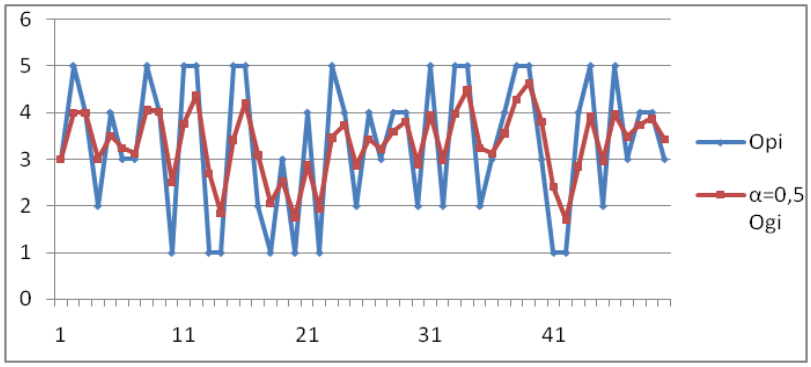


Illustration 7.14: General Opinion simulation example

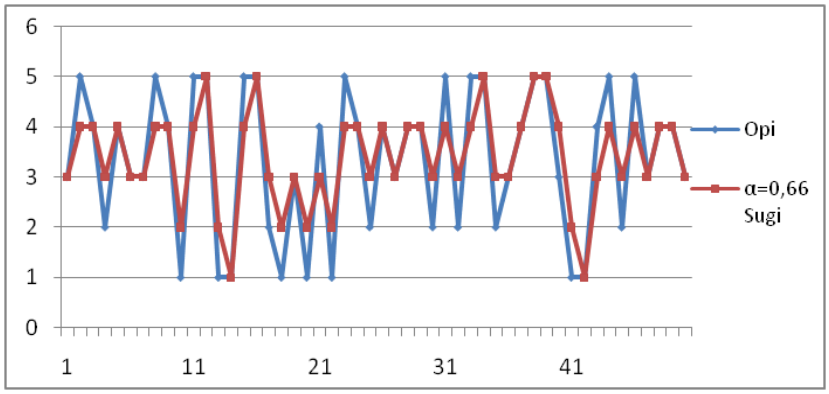


Illustration 7.15: Suggestion simulation example

The Simulation Record Table figuring the simulation through the statistical measures used is available for consultation in annex A *"Statistical Data"*.

7.4.3.5 Data analyses

From the set were extracted two information tables ready to be interpreted.

The first study was conceived to see the average difference in every record between the most regular general opinion and suggestion to the average personal opinion. The difference gives a sense of how close the general opinion and suggestion generated by α is to the average personal opinion.

	α = *0,5*		α = *0,33*		α = *0,66*	
	Og	*Sug*	*Og*	*Sug*	*Og*	*Sug*
Op	0,59	0,62	0,40	0,57	0,77	0,80

Table 7.7: Results given by the first study

The second study shows the direct difference between the personal opinion average mean to general opinion and suggestion. Greater the value of deviation, greater chance of the general opinion and suggestion not be synchronized with the most prevalent personal opinion.

	α = *0,5*		α = *0,33*		α = *0,66*	
	Og	*Sug*	*Og*	*Sug*	*Og*	*Sug*
Op	0,01	0,00	0,02	0,04	0,00	0,06

Table 7.8: Results given by the second study

The first study compare the performance of the α by every simulation record while the last evaluates the performance globally by processing the stored results of the 50 simulations in regard to the average value.

The α = *0,5* updates the current general opinion by making it lean toward the new opinions with a neutral pace. This value neither highlights the general opinion nor accentuates the contribution of new personal opinions.

To α = *0,33* is emphasized the value of the current general opinion, making the progression to new opinion slower. A factor α = *0,33* can of reduce the impact of random extreme opinions which could distort the estimate of a realistic general opinion.

The α = *0,66* facilitates the adoption of new personal opinions. This method is ideal when sudden changes in the personal opinions occur periodically spaced, making the general option harmonize faster. If the behaviour of the personal opinions shifts constantly around extreme values makes the general opinion too sensible to the changes.

In the table 7.7 we can see that the α = *0,33* the better performance.

In the second table, α = *0,5* had the lowest deviation score (*Og = 0,01; Sug = 0,00*) making the values returned by his application usually more close to the personal opinion tendency.

To a lower smoothing factor the general opinion and the suggestion tends to be more close to the average personal opinion, while they seem to suffer from worse effectiveness in keeping pace with the most recent personal opinion.

By the objective of the analyses both α = *0,5* and α = *0,33* fared generally better than the alternative, having the lower deviation and the smaller difference value from the average personal opinion. This performance was already expected at the beginning of the simulation.

The combination of these two factors means that lower α gets values more close to the most common personal opinion at a very satisfactory rate.

The smoothing value should be updated in the future to increasingly adapt to the behaviour of the population:

- If the population tends to have a very regular set of opinions punctuated with sporadic extreme opinions α should decrease.
- If the population shifts to new set of opinions periodically α should increase;

For now the α will be set with a value of **0,45**.

7.5 Proximity Update

Proximity is a form of correlation between a pair of elements from the same data source.

$$Proximity = P(Elem(Type(y,ds)\ i, v), Elem(Type(z,ds), j), v), if(y = z) \longrightarrow i \neq j;$$

equation 7.11: Proximity

The correlation informs how frequently two elements are associated in tasks destined to monitor events simultaneously in a group of elements.

Whenever a task involving a group elements suffer a type of action (creation, change or elimination) by the crew member, the strength of the proximity between the affected elements of the group must be updated.

7.5.1 Proximity update by creation

When a task with a multiple elements to be monitored is created the proximity each pair of elements is reinforce according to:

$$P(Elem(Type(y, ds)\ i, v), Elem(Type(z, ds), j), v)_k = P(Elem(Type(y, ds)\ i, v), Elem(Type(z, ds), j), v)_{k-1} + 1$$

equation 7.12: Proximity update by creation

7.5.2 Proximity update by elimination

When a task submitted by the user is deleted by him the relationship must correspond to the decline:

$$P(Elem(Type(y, ds)\ i, v), Elem(Type(z, ds), j), v)_k = P(Elem(Type(y, ds)\ i, v), Elem(Type(z, ds), j), v)_{k-1} - 1$$

equation 7.13: Proximity update by deletion

7.5.3 Proximity update by change

Proximity update by change applies the combination of the methods found previously .

For the new elements added to the group the new proximity see equation 7.12.

For the elements removed from the group the relation is updated equation 7.13.

7.6 Conclusions

The Portal DOV Interface Agent implementation went more efficiently than the previous phase, thanks to the vital insights provided earlier. The insights which allowed this step to progress relatively well, derived from by the process of collection of requirements and the three models completed during the analysis phase from Gaia. Mainly the environmental model and the role model allowed to grasp the requirements in two important ways: first in terms what the agent would need to perform his tasks and what he would do to achieve his goals. By analogy these qualities related data structures we would interact and what the general processes needed to achieve his goals. Although the models were fairly high level with little the details about the how, they gave the blueprints for the definition of the data model adopted and important aspects about the general behaviour of the agent.

Visual Studio 2008 during development proved to be fairly adequate with all the functionalities and features present, although the official documentation not always would prove to be very useful when facing one or another intrinsic problem found to the platform during development.

8 Conclusions and Future Work

The definition of the Portal DOV Agent architecture through the application of the Gaia methodology showed maladjusted. Gaia is especially oriented for multi-agent environments with a bigger degree of complexity, and thus introduced, in my opinion, redundant steps or unframed processes given the relative simplicity of Portal DOV Interface Agent. The application of the TROPOS generated antagonism due to ambiguity in the stakeholders in the present context. Although the application of Gaia stalled, its application had its revenues as the environment and preliminary role model along with the some concepts from early-requirements clarified the agent's process of actuation and allowed to apprehend the concepts of necessary data for its implementation and operation of the agent.

By implementing Portal DOV Interface Agent, an interface agent with ability to make suggestions supported by the professional category of the owner we hope to improve the dissemination of crucial information to each professional category. The key trait is lack of direct intervention by the official channels. All the crew members help to define what is important to them at a time and indirectly share that knowledge or opinion between them in the form of general opinion.

Because the gathering and processing of the tasks is continuous this makes current suggestions always sensitive to the needs of the users.

Two immediate experiments can provide more accurate results about the actual performance of the agent, being the first one a more extended use of the interface agent by the crew members and evaluation of the crew members's feedback about the agent behaviour and utility.

With time, a more precise perception about the general behaviour of the crew members can become more evident and so help to locate a smoothing factor more efficient. The smoothing factor will be gradually adjusted according to the variations.

Alternatively, a different approach to the calculation of suggestion can be adopted. The current data structure supports the implementation of different methods to find a general opinion by different ways because it stores the data concerning all the past tasks. The past data is a enormous source of information for the application of different statistical methods. An interesting test would be to evaluate the difference in performance between the exponential moving average and suitable method of comparison like mean average or preferably the weighted arithmetic mean. Since the data structure allows with little effort the implementation of different methods for the calculation of relevance we have greater flexibility and a greater chance of adaptability of the agent to current and future patterns of crew member behaviour.

One interesting path would be to expand the number of present category views applied to extract suggestions and so get increasingly more detailed users profiles. While the number of viewpoints to evaluate opinions may grant more insight about the nature of each user and consequently more adequate answers, it comes at a cost. The cost would derive directly of the rate of growth of the data structure discussed in chapter 7.2.5.3 .

One future course of action could be to broaden the number of data sources monitored. Currently the Portal DOV Interface Agent is expressly conceived for crew member in regard to the interaction with the two data sources, documentation and roster. Yet there are more information sources present in Portal DOV with the possibility of exploration. BIDS can be seen as a data source made available through Portal DOV to let the crew member choose preferential pairing and/or day offs. The interface agent could represent the crew member in his need to evaluate the feasibility of his request without the interference of Portal DOV.

By the many reasons and considerations pointed in this chapter we can see that the interface agent is a benefit working in both ways. By the perspective of crew member, promotes the timely acquisition of relevant information granting the user with more independence and autonomy. To Portal DOV, the major improvement comes with clarification of the notion of crucial content throughout the professional categories. This allows a more efficient diffusion of information without the direct interference of Portal DOV or other entities.

References

[Bresciani03]Paolo Bresciani. Fabrizio Sannicolò. Requirement analysis in TROPOS: a self referencing example. J. Mller, H. Tianfield, R. Unland, R. Kowalszyk, editors. In Agent Technologies, Infrastructures, Tools, and Applications for e-Services. 2003. Springer-Verlag.

[Bresciani04]Paolo Bresciani. Paolo Giorgini. Fausto Giunchiglia. John Mylopoulos. Anna Perini. TROPOS: An Agent-Oriented Software Development Methodology. In Journal of Autonomous Agents and Multi-Agent Systems. May 2004. Kluwer Academic Publishers.

[Castro06] António Castro 2006, "Designing a Multi-Agent System for Monitoring and Operations Recovery for an Airline Operations Control Centre", Faculty of Engineering, University of Porto and LIACC, 2006

[Croft97] David Wallace Croft 1997, "Intelligent Software Agents: Definitions and Applications", 1997, [online] http://alumnus.caltech.edu/~croft/research/agent/definition/ [consulted on December, 2007,1 4]

[CSLangSpec07] C# Language Specification Version 3.0, Microsoft Corporation, 2007, pp. 1, [online] http://msdn.microsoft.com/en-us/vcsharp/aa336809.aspx .[consulted on May, 2008, 18]

[Date03] C. J. Date, 2003, "What First Normal Form Really Means", pp. 127-8 June 2003

[Dateo3b] C. J. Date, "Introduction to Database Systems, An", Addison Wesley; 8 edition, 2003

[FrankGraess96] Stan Franklin and Art Graesser 1996, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents" Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996, [online] http://www.msci.memphis.edu/~franklin/AgentProg.html

[Giorgini04]Paolo Giorgini. Manuel Kolp. John Mylopoulos. Marco Pistore. The Tropos Methodology: an overview. In Methodologies And Software Engineering For Agent Systems. 2004. Kluwer Academic Publishing.

[GuimCabral97] Guimarâes, Rui Campos. Cabral, José Sarsfield. ESTATÍSTICA, McGraw-Hill de Portugal, 1997

[Maes97a] Pattie Maes 1997, "What is an Agent?", Software Agents Tutorial, MIT Media Laboratory, [online] http://web.media.mit.edu/~pattie/CHI97/sld005.htm [consulted on December, 2008, 15]

[Maes97b] Pattie Maes 1997, "Interface Agent <-> User", Software Agents Tutorial, MIT Media Laboratory, [online] http://web.media.mit.edu/~pattie/CHI97/sld058.htm [consulted on December, 2007, 16]

[Maes94] Pattie Maes 1994, “Social interface agents: Acquiring competence by learning from users and other agents”, Etzioni, O., editor, Software Agents - Papers from the 1994 Spring Symposium (Technical Report SS-94-03), pp. 71-78, AAAI Press, 1994.

[Mingers95] J.C. Mingers 1995, ‘What is the distinctive nature and value of IS as a discipline?’,The Systemist, Vol.17, No. 1, 1995, pp. 18–22.

[Nwana96] Hyacinth S. Nwana 1996, “Software Agents: An Overview Knowledge” Engineering Review, Vol. 11, No 3, pp.1-40, Sept 1996. Cambridge University Press, 1996.[online] http://agents.umbc.edu/introduction/ao/ [consulted on December, 2007, 12]

[Reis03] Luís Paulo Reis, Coordenação em Sistemas Multi-Agent: Aplicações na Gestão Universitária e no Futebol Robótica, Tese de Doutoramento em Eng. Electrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto, July, 2003

[RussNorvig03] Stuart Russell and Peter Norvig 2003, “Artificial Intelligence A Modern Approach”, Second Edition, pp. 32. Prentice Hall, 2003.

[UKAIS99] UK Academy for Information Systems. THE DEFINITION OF INFORMATION SYSTEMS, July 1999, [online] http://www.turningcourse.com/ukais/isdefn.pdf [consulted on January, 2008, 28]

[WardPepp02]John Ward. Joe Peppard. Strategic Planning for Information Systems, John Wiley & Sons; 3rd edition. 2002

[WoolJenn95] Michael Wooldridge and Nicholas Jennings 1995, “Intelligent Agents: Theory and Practice”, The Knowledge Engineering Review 10 (2), pp. 115-152, 1995.

[Zambo01] Zambonelli, F., Jennings, N. and Wooldridge, M. (2001) “Organisational rules as an abstraction for the analysis and design of multi-agent systems”, International Journal of Software Engineering and Knowledge Engineering, Vol. 11, No. 3, pp.303–328.

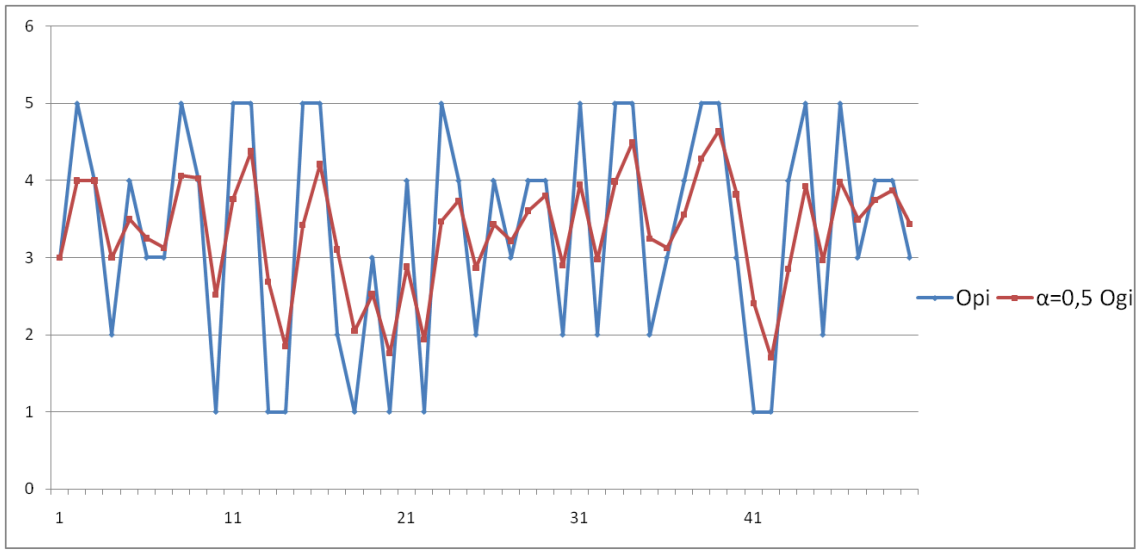
[Zambo03] Franco Zambonelli, Nicholas R. Jennings and Michael Wooldridge, 2003, “Developing Multiagent Systems: The Gaia Methodology”, ACM Inc., ACM Transactions on Software Engineering and Methodology, Vol. 12, nº3, 2003, pp. 3 41, [online] http://users.ecs.soton.ac.uk/nrj/download-files/tosem03.pdf .[consulted on March, 2008, 16]

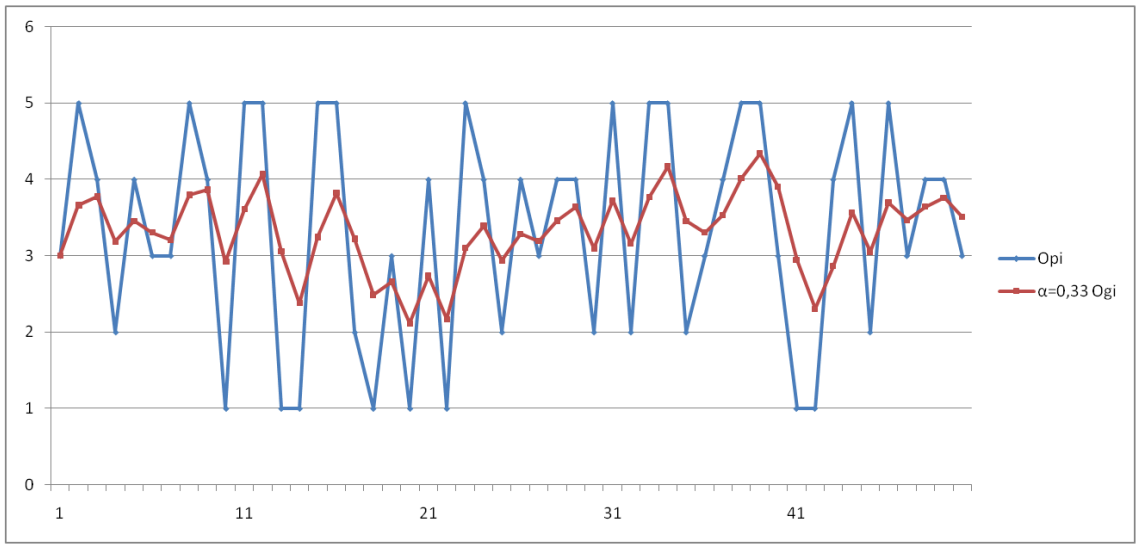
Anexo A: Statistical Data

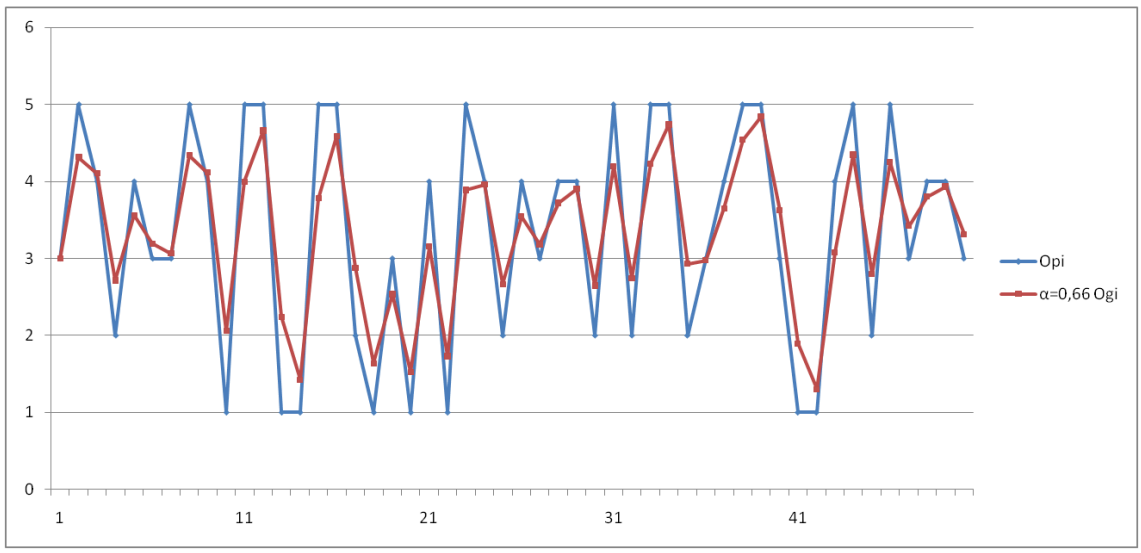
Random Sampling

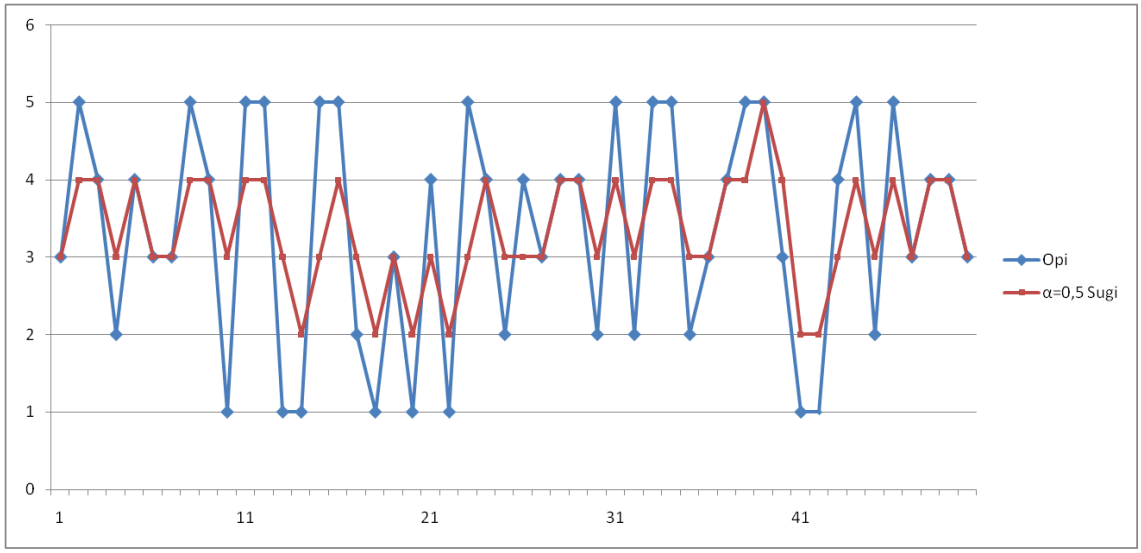
		General Opinion			Suggestion		
i	Opi	α=0,5	α=0,33	α=0,66	α=0,5	α=0,33	α=0,66
0	###	3	3	3	###	###	###
1	3	3	3	3	3	3	3
2	5	4	3,66	4,32	4	4	4
3	4	4	3,7722	4,1088	4	4	4
4	2	3	3,187374	2,716992	3	3	3
5	4	3,5	3,455541	3,563777	4	3	4
6	3	3,25	3,305212	3,191684	3	3	3
7	3	3,125	3,204492	3,065173	3	3	3
8	5	4,0625	3,79701	4,342159	4	4	4
9	4	4,03125	3,863997	4,116334	4	4	4
10	1	2,515625	2,918878	2,059554	3	3	2
11	5	3,757813	3,605648	4,000248	4	4	4
12	5	4,378906	4,065784	4,660084	4	4	5
13	1	2,689453	3,054075	2,244429	3	3	2
14	1	1,844727	2,376231	1,423106	2	2	1
15	5	3,422363	3,242074	3,783856	3	3	4
16	5	4,211182	3,82219	4,586511	4	4	5
17	2	3,105591	3,220867	2,879414	3	3	3
18	1	2,052795	2,487981	1,639001	2	2	2
19	3	2,526398	2,656947	2,53726	3	3	3
20	1	1,763199	2,110155	1,522668	2	2	2
21	4	2,881599	2,733804	3,157707	3	3	3
22	1	1,9408	2,161648	1,73362	2	2	2
23	5	3,4704	3,098304	3,889431	3	3	4
24	4	3,7352	3,395864	3,962407	4	3	4
25	2	2,8676	2,935229	2,667218	3	3	3

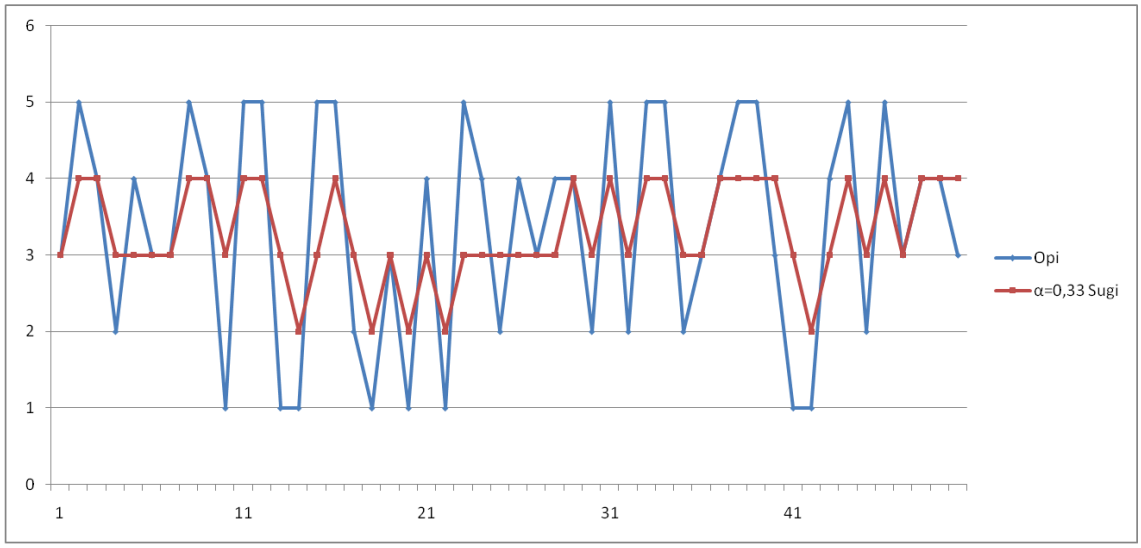
		Random Sampling					
		General Opinion			Suggestion		
i	Opi	α=0,5	α=0,33	α=0,66	α=0,5	α=0,33	α=0,66
26	4	3,4338	3,286603	3,546854	3	3	4
27	3	3,2169	3,192024	3,18593	3	3	3
28	4	3,60845	3,458656	3,723216	4	3	4
29	4	3,804225	3,6373	3,905894	4	4	4
30	2	2,902112	3,096991	2,648004	3	3	3
31	5	3,951056	3,724984	4,200321	4	4	4
32	2	2,975528	3,155739	2,748109	3	3	3
33	5	3,987764	3,764345	4,234357	4	4	4
34	5	4,493882	4,172111	4,739681	4	4	5
35	2	3,246941	3,455315	2,931492	3	3	3
36	3	3,123471	3,305061	2,976707	3	3	3
37	4	3,561735	3,534391	3,65208	4	4	4
38	5	4,280868	4,018042	4,541707	4	4	5
39	5	4,640434	4,342088	4,84418	5	4	5
40	3	3,820217	3,899199	3,627021	4	4	4
41	1	2,410108	2,942463	1,893187	2	3	2
42	1	1,705054	2,30145	1,303684	2	2	1
43	4	2,852527	2,861972	3,083252	3	3	3
44	5	3,926264	3,567521	4,348306	4	4	4
45	2	2,963132	3,050239	2,798424	3	3	3
46	5	3,981566	3,69366	4,251464	4	4	4
47	3	3,490783	3,464752	3,425498	3	3	3
48	4	3,745391	3,641384	3,804669	4	4	4
49	4	3,872696	3,759727	3,933588	4	4	4
50	3	3,436348	3,509017	3,31742	3	4	3

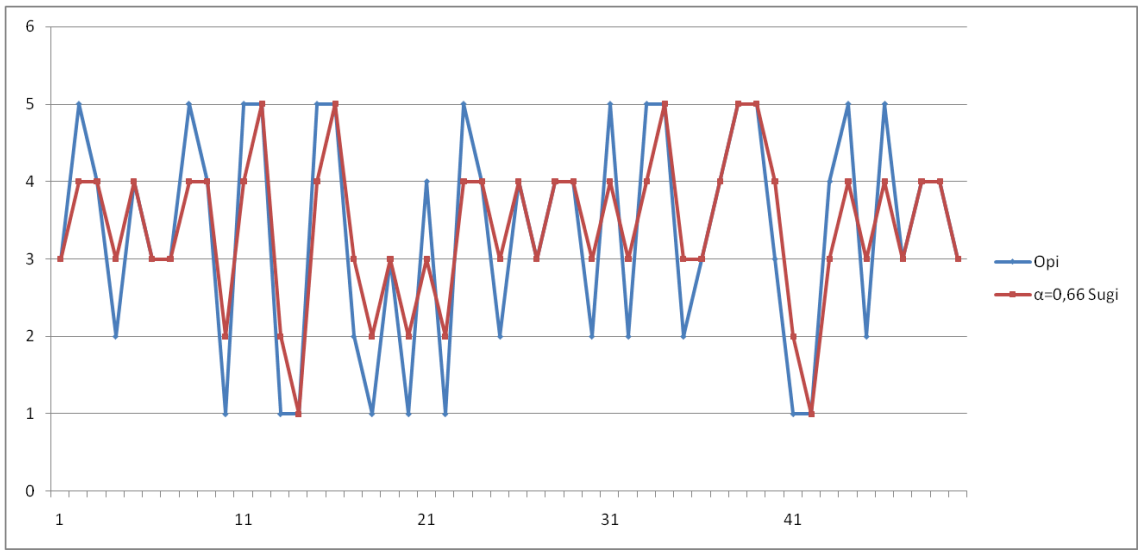

6
5
4
3
2
1
0
1
11
21
31
41
Opi
α=0,5 Ogi


6
5
4
3
2
1
0
1
11
21
31
41
Opi
α=0,33 Ogi


6
5
4
3
2
1
0
1
11
21
31
41
Opi
α=0,66 Ogi


6
5
4
3
2
1
0
1
11
21
31
41
Opi
α=0,5 Sugi


6
5
4
3
2
1
0
1
11
21
31
41
Opi
α=0,33 Sugi


6
5
4
3
2
1
0
1
11
21
31
41
Opi
α=0,66 Sugi