Universidad Carlos III de Madrid
Escuela Politécnica Superior
Bachelor in Computer Science and Engineering

BACHELOR THESIS

Circumstantial Knowledge Management for Human-Like Interaction

Author: Alejandro Baldominos Gómez
Tutor: Francisco Javier Calle Gómez

Leganés. June, 2012

Email
abaldomi@inf.uc3m.es

Phone
+34 91 624 9114

Address
Advanced Databases Group (LaBDA)
Computer Science Department
Universidad Carlos III de Madrid
Avda. de la Universidad, 30
28911 Leganés (Madrid) - Spain

Please, cite this thesis as
Baldominos, A. (2012). *Circumstantial Knowledge Management for Human-Like Interaction*. Bachelor thesis, Universidad Carlos III de Madrid.

BACHELOR THESIS

Circumstantial Knowledge Management for Human-Like Interaction

Author: Alejandro Baldominos Gómez
Tutor: Francisco Javier Calle Gómez

THE EXAMINING BOARD

President: Ricardo Colomo Palacios
Secretary: Raúl Arrabales Moreno
Member: Manuel Carretero Cerrajero

After the defense of the **Bachelor Thesis**, taking place in **Escuela Politécnica Superior** of **Universidad Carlos III de Madrid** (Leganés) on July 3rd, 2012, the examining board agrees to confer the next **GRADE**:

This page has been intentionally left blank.

It's difficult to be rigorous about whether a machine really "knows", "thinks", etc., because we're hard put to define these things. We understand human mental processes only slightly better than a fish understands swimming.
John McCarthy (1927-2011)

We can only see a short distance ahead, but we can see plenty there that needs to be done.
Alan Turing (1912-1954)

This page has been intentionally left blank.

Agradecimientos

Durante la realización de un proyecto como éste uno se siente constantemente apoyado. El interés y la colaboración de las personas que me rodean son sin lugar a dudas un incentivo para la realización del trabajo, y es ahora mi intención devolver una pequeña parte de aquellos en forma de agradecimiento.

En primer lugar, debo agradecer a mis padres su interés permanente en mi trabajo, a pesar de que por cuestiones técnicas y de idioma, sé que serán incapaces de entender una sola palabra del mismo. A veces, la pregunta recurrente de cuántas páginas había escrito ese día me hacía ver realmente mi progreso con el trabajo.

Un agradecimiento muy especial se lo dirijo a Javi, por todas esas reuniones de *'diez minutos'* que en realidad acababan durando horas, reuniones en las que de algún modo me ha conseguido introducir la idea de que todo es tremendamente sencillo, sin importar el número de dimensiones a considerar. También, aunque hayamos tenido menos contacto, debo agradecer a Loli su disponibilidad para atenderme con cualquier duda que haya podido tener.

Durante cuatro años de carrera, a uno le da tiempo a conocer a muchos profesores, pero no todos consiguen que te cuestiones tu propia forma de pensar. Gracias, Gonzalo, por hacerme ver que detrás de cada idea, por evidente que hoy nos resulte, alguien en algún momento de la Historia tuvo que luchar para poder establecerla.

También quiero agradecer a mis compañeros del Laboratorio de Bases de Datos los buenos ratos que me hacen pasar a diario. Aunque cada vez dedico más horas en el laboratorio, habéis conseguido que cada vez la jornada se me haga más corta. Aprovecho para agradecer a María y Chumo el tiempo que dedicaron para participar en mi evaluación. Espero que pasárais un buen rato con ella. Y por último, me gustaría dedicar un agradecimiento muy especial a Espe y a Gara, no sólo por participar en la evaluación, sino por dejar lo que estuvieran haciendo para ayudarme con cualquier duda que he podido tener en cualquier momento. Muchas gracias chicas.

Entre mis compañeros y amigos de la Universidad me gustaría dar las gracias a aquellos que de un modo u otro han colaborado conmigo en la realización de este trabajo: a José Luis, por sus visitas tan inspiradoras al laboratorio; a Carlos, por su constante interés en el trabajo y por compartir conmigo sus fantásticas ideas; a Sara, por preguntarme por

mi trabajo constantemente; a Adela por soportar mis explicaciones (entretenidas, espero) sobre mi trabajo; a Josito, por compartir conmigo su estrés diario con el TFG (¡a pesar de que finalmente lo has acabado antes que yo!) y a Irene por su utilísima contribución como socióloga.

Y por último, agradezco de un modo especial a Nerea, por la excelente labor que ha realizado reportándome erratas en el trabajo y compartiendo su punto de vista. Pero por encima de todo, por haber sido capaz de aguantar estoicamente, durante estos años, mi *fingida* prepotencia.

Gracias.

Abstract

This project focuses on the circumstantial aspects of human-like interaction. Its purpose is in the first place to design and develop a general-purpose situation model, which would enhance any natural interaction system by providing knowledge on the interaction context, and secondly to implement an intuitive management tool to edit the knowledge base of this situation model.

The development of both the model and the edition tool has followed the usual processes of software engineering: requirements elicitation and specification, problem analysis, design of a solution, system implementation and validation. In more specific terms, an spiral lifecyle composed of three phases was followed, which is a convenient approach in research projects as all the system requirements might not be known from the very beginning.

After the implementation was completed, an evaluation was carried out for observing the advantages of the edition tool over the manual edition of the model knowledge. The results of this evaluation showed that the tool provides a mechanism for the model management which is significantly faster and more accurate than the manual edition. Moreover, a subjective survey also revealed that the experiment subjects preferred the edition tool, as they considered it to be more comfortable, more intuitive, more reliable and more agile.

The resulting general-purpose situation model and the edition tool are a significant contribution to the state of the art, as the previously existing situation models were *ad-hoc* models, i.e., models implemented for supporting an specific system, and their knowledge bases were edited manually.

Finally, this work will be applied in a research project funded by the Spanish Ministry of Industry (CADOOH, TSI-020302-2011-21), and for this reason some of the future works observed in this document will be executed in the coming months. A copy of the Cognos toolkit, including the edition tool developed within this project, is available in the next site: `http://labda.inf.uc3m.es/doku.php?id=es:labda_lineas:cognos`.

This page has been intentionally left blank.

Keywords

Circumstantial Knowledge, Situation Model, Natural Interaction, Context, Context-Aware System, Knowledge Base, Knowledge Management, Management Tool.

This page has been intentionally left blank.

Contents

This page has been intentionally left blank.

List of Figures

List of Tables

This page has been intentionally left blank.

Chapter 1

Introduction

In recent years, personal computers have evolved so much that we can now find them in many different forms, such as smartphones, tablets or navigation systems.

As most of the people are gaining access to those devices, new interfaces are required so that the users can interact with systems in a simple way, thus not requiring any previous training or specific knowledge. Moreover, these interfaces must be useful and accesible to all the users, regardless of their condition or their technical capabilities.

The aim of human-like interaction is to provide interfaces which imitate the way in which humans interact with each other. With this purpose in mind, some particular knowledge about the context may assist the system to provide a more elaborated interaction, thus achieving a more natural and realistic way of communication with a computer system.

1.1 Theoretical Background

In the first place a theoretical background for this project is provided. We will focus on briefly describing what the circumstantial knowledge is and how a situation model can enhance a human-like interaction system in order to provide a more complete and realistic experience.

Circumstances are defined as facts or conditions which are relevant to a particular event or action. Regarding an interaction process, we can define the circumstantial knowledge as the set of knowledge and skills regarding the context, and therefore the circumstances, of the interaction.

The circumstantial knowledge can be classified in different aspects. According to Gee taxonomy [Gee, 1999], the context in a communicative social interaction comprises the following components: a **semiotic** aspect regarding the language used in the communication process; an **activity** (or operative) aspect concerning the transactions (the major activi-

ties or tasks) underlying the communication; a **material** aspect respecting to the physical situation of the action, that is, mainly the spatio-temporal context, including people and objects taking part on the interaction, yet also other environmental eventualities such as noise, weather, etc; a **political** aspect with regard to the roles or status of each person taking part in the communication process; and a **sociocultural** aspect observing the influence on the interaction of the specific social and cultural conditions underlying the interaction.

Once we have introduced what the circumstantial knowledge and its components are, we can delve into the concept of *situation model*. In a first approach, it should be stated that a situation model formalizes, stores and processes the circumstantial knowledge for the communication process. The inclusion of a situation model in a human-like interaction system [Rivero et al., 2007] provides some advantages:

- Supports more knowledge influence in the produced interaction, thus allowing the system to adapt its operations to the current context and enhancing the naturality of the communication.
- Enables the use of circumstantial information for filtering other components' knowledge bases, thus reducing ambiguity and resolving indexicals, and consequently increasing the efficiency and efficacy of the system.
- Provides mechanisms for situation triggering, i.e., to perform a certain action when some spatio-temporal conditions are fulfilled.
- Provides new circumstances-related functionalities such as guidance to a certain situation from the current one or predictions on an object situation in the future.

However, the situation model is interesting even when not included within a human-like interaction system. At the very least, it provides context-awareness capabilities to any system, thus empowering it. For instance, we can imagine a navigation system which not only plans a route to the user destination, but it does it taking into account ambiental factors, time conditions and the user preferences; thus enriching its functionality by means of a situation model.

1.2 Project Objectives

In order to achieve human-like interaction, the human behavior in the communication process must be studied, formalized and stored in a form that can be processed by a computer system. This task includes the formalization and storage of knowledge regarding the circumstantial aspects of the interaction.

The purpose of this project is to ease the implementation of specific situation models for a human-like interaction system by providing edition tools able of feeding its knowledge

bases. From the five different components of the circumstantial knowledge which we have previously described, the scope of the project is restricted only to the material aspect, i.e., the spatio-temporal component of the contextual information. However, a partial support of the remaining aspects will be provided by means of user-defined situations.

This situation model will be implemented over a spatio-temporal database [Cuadra et al., 2008], as it is able to store the location and form description of objects and its evolution through time.

Additionally, a tool will be developed to manage the circumstantial knowledge stored in the database. This tool will provide means to edit the spatial network graph and to define objects associated to certain spatial and temporal conditions.

Finally, another tool will be developed in order to perform a simulation over the situation model. The main aim of this simulator is to show the power of the situation model itself, before the integration with other modules of the human-like interaction architecture takes place.

Some scenarios where the implemented situation model can be applied are described in appendix A.

1.3 Project Context

Figure 1.1 shows a model-based cognitive architecture for a natural interaction system proposed by [Calle, 2004]. In this architecture, the Presentation Model and its associated interface components parse the user inputs and synthesize the outputs. Different technologies can be implemented for this module, such as speech recognition and synthesis, gestures recognition, a 3D character, etc. Supporting their interpretation in the interaction system, the Ontology deals with the set of concepts and the relations among them which are referenced in the communication process.

The Dialogue Model processes the user communicative acts and develops its own acts and the Discourse Generator manages the system tasks during the interaction process and produces the contents of the interaction.

Meanwhile, the Emotional Model operates the emotional state, the User Model manages the interlocutor features, the Session Model stores the evolution of the interaction and the Self-Model manages the system goals.

Finally, the Situation Model, which is the aim of this project, manages the contextual information of the interaction.

All these models work over a multi-agent architecture, thus acting in an independent but collaborative way. The advantages of this architecture are presented in [Izquierdo, 2011].

The presented cognitive architecture is currently being applied in the Cognos project

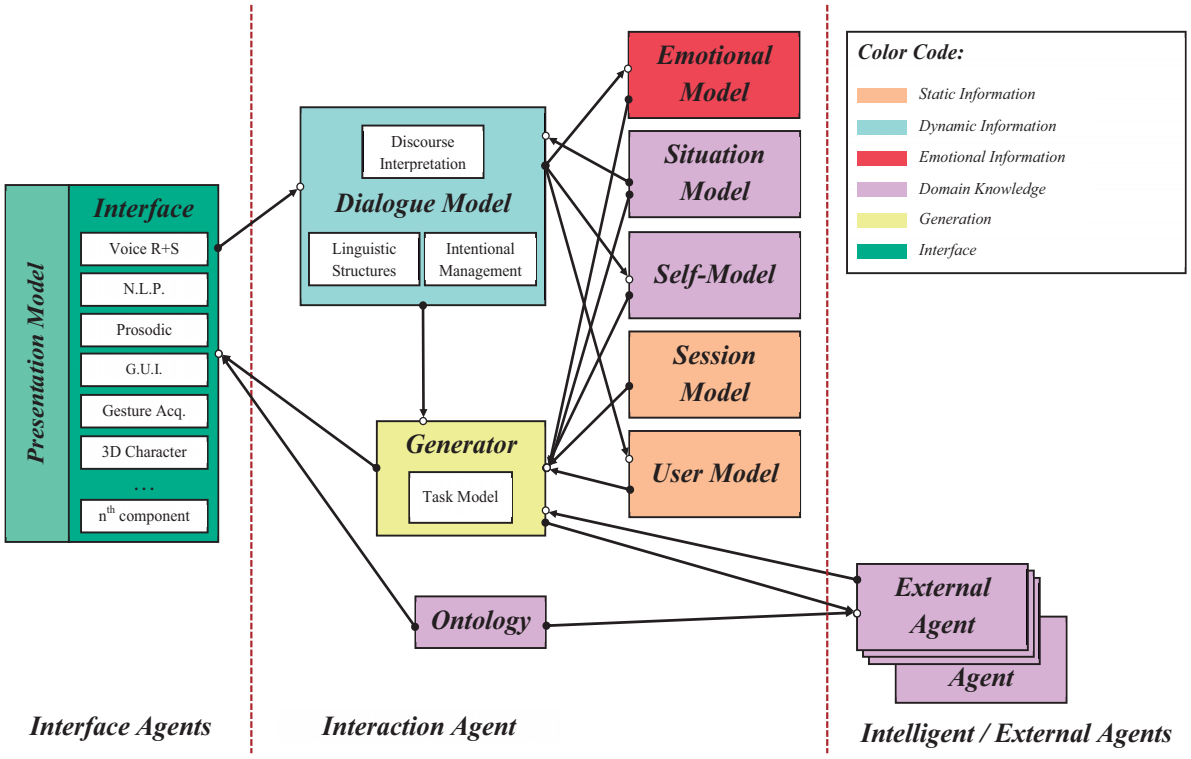


Figure 1.1: Cognitive architecture for a natural interaction system

[Calle et al., 2010, Calle et al., 2011], which is carried out at Universidad Carlos III de Madrid by the Advanced Databases Group (LaBDA) group (`http://labda.inf.uc3m.es`) of the Computer Science Department. The current work could eventually be embodied into the Cognos toolkit.

1.4 Document Structure

In the first place, it is adequate to introduce the state of the art of the concepts, technologies and techniques related to the aim of this project, such as ubiquitous computing, context-aware systems, context models, natural interaction and situation models, as well as presenting some applications. Chapter 2 contains this information.

Chapter 3 provides the description for the project. The main aim of that chapter is to formalize the system functionality through its requirements specification. Once the user requirements are formalized, a feasibility study is performed to ascertain if the goals and requirements are realistic or not, given some previous work on the field and known the technical and environmental background. The chapter also includes the definition of

validation tests, to cover all the user requirements. Finally, a planning is elaborated which describes the development methodology as well as an schedule and a cost projection for the whole development process.

Chapter 4 provides a high-level design of the system, both in terms of the physical and functional architecture, and 5 delves into this design, describing the system architecture with enough detail to serve as a guide for its implementation.

After the implementation phases are carried out, chapter 6 provides the results of the execution of the validation tests, in order to determine whether the developed system meets the user requirements; and proposes and performs a comparative system evaluation over manual edition.

Finally, chapter 7 provides conclusions on the project, and also provides some lines of future work in the research and development of the project.

After the main content, the document includes five appendices. Appendix A presents different scenarios where the situation model could be useful, whereas appendix B describes some ethical considerations in the project. Appendix C presents the original data gathered during the evaluation process. Finally, appendices D and E contains an installation guide and a user manual respectively.

This page has been intentionally left blank.

Chapter 2

State of the Art

The purpose of this chapter is to introduce the state-of-the-art for the project. Sections 2.1 to 2.6 aim to introduce key concepts which are related to the topic of this project, as well as to provide a historical overview on how these concepts evolved eventually leading to the appearance of natural interaction and situation models. Although there are few practical applications of situation models, section 2.7 describes systems where situation models have been successfully applied. Finally, some conclusions are provided on the state of the art.

2.1 Ubiquitous Computing

In the late 1980s, the term *ubiquitous computing* was coined to refer to a computing paradigm in which computers are everywhere and they turn out to be invisible for humans, meaning that the user can focus on the task he is performing instead of focusing on the way he interacts with the system.

Anecdotically, this term was first used by Steve Jobs in 1987, with a different meaning that the one that was later assigned to it: he referred to the *Apple II* computer as an ubiquitous computing resource that could be found everywhere on campus due to its reliability and low cost [Potts, 1987].

One year later, in 1988, Mark Weiser envisioned the concept of ubiquitous computing that has reached our days; however, it was not until 1991 that an article about it was published [Weiser, 1991]. The main idea beyond ubiquitous computing, as described in this article, is to conceive a new way of thinking about computers that focuses in the human world and vanishes computers into the background, that is, the computer technology is so spread out that people use it without thinking about the technology itself, but focusing in their own goals.

In order to achieve a real ubiquitous computing, Weiser identifies two key issues: location and scale [Weiser, 1991]. Regarding the first one, a computer that knows where it is

located can adapt its behavior to the circumstances in significant ways. Meanwhile, the issue of scale is related to the fact that computers would have different sizes in order to suit a particular task.

Delving into the issue of scale, Weiser proposed three different types of components for ubiquitous systems, each of them with different sizes and purposes. The smallest components are called **tabs**, which are sized like a post-it or ID card. Slightly bigger devices would be **pads**, sized like a sheet of paper (A4), they do not need to be carried out with the user, but instead they will stand as *scrap computers*. The last and biggest devices are called **boards**, which can serve as video screens, digital whiteboards or bulletin boards, etc. Weiser specifies three technologies which are required for supporting ubiquitous computing: cheap computers with convenient displays, software for ubiquitous applications and a network to allow the communication among the different systems.

Different computing eras have been identified according to the number of computers per person [Weiser and Brown, 1996]. Within this taxonomy, ubiquitous computing is historically placed as a new computing paradigm which follows the personal computing. The different computing paradigms or eras can be summarized as follows:

- The mainframe era, in which many people (usually experts) shared one computer mainly for scientific purposes.
- The Personal Computer (PC) era, in which each person has its own computer. In 1984, the number of people using a PC surpassed those using shared computers [IDC, 1996]. Since then, the appeareance of laptop computers, Personal Digital Assistants (PDAs) and mobile phones have contributed to the establishment of the PC era and the evolution to the ubiquitous computing era.
- The ubiquitous computing era, in which each person owns many computers which are interconnected.

A few years later, in 1994, another term closely related to ubiquitous computing appears: this term is *pervasive computing*. It was first used to refer to Novell strategies for connecting people to people and to information any place at any time [Schofield, 1994]. After some years of decline, the term reappered in 1998 in IBM's post-PC world [Lohr and Markoff, 1998], this time meaning that computers were *everywhere*. The main objective during that time was that computer devices could be everywhere and that they allowed their owners to be connected to the Internet in any place at any moment.

Despite the fact that the terms *ubiquitous computing* and *pervasive computing* are mostly used as synonyms today, their original meanings were significantly different [Ronzani, 2009]: while the term *ubiquitous computing* referred to an Human-Computer Interaction (HCI) style which turns out the system to be invisible to the user, which is obtained through miniaturising some computer devices and reducing

their costs; the term *pervasive computing* focused on the spread of portable and handheld systems, which allowed the user to be connected to the Internet everywhere.

Nowadays, the appearance of many handheld computer devices such as smartphones and tablets are taking us to a progressive achievement of ubiquitous computing. However, the current situation still falls short of the original vision introduced by Weiser, in which computers are invisible to humans, who can concentrate on the tasks they are performing. Moreover, the idea behind ubiquitous computing introduces new challenges; particularly, the computer devices can be located everywhere and consequently some knowledge about the environment might support their adaptation to circumstances.

2.2 Ambient Intelligence

In 1999, another key term related to the concept of ubiquitous computing arises: this term is *ambient intelligence* and appears for the first time in a paper written by the director of Philips about the digital home [Jones, 1999]. Two years later, in 2001, the European Comission designs a roadmap for research in ambient intelligence [Ducatel et al., 2001], in which are also described four scenarios on how ambient intelligence could be applied to everyday life in 2010 (from the current point of view, those scenarios turned out to be too ambitious).

The basic idea behind ambient intelligence is that by incorporating some technology to an environment, a system can be built for executing actions that will provide some benefits to the users in that environment. The system, which would be transparent to the user, would decide by itself which actions to take by sensing the environment and the users in it and processing this information.

Some features which are commonly expected in ambient intelligence technologies requires them to be sensitive, responsive, adaptive, transparent, ubiquitous and intelligent [Cook et al., 2007]. Thus, it can be realized how closely related is ambient intelligence with other concepts such as ubiquitous computing, pervasive computing and context-aware systems. In order to be able to provide these features, the IST Advisory Group [Ducatel et al., 2001] described five key technological requirements: the need of very unobstrusive hardware, which can be obtained through miniaturisation; a seamless web-based communications infrastructure to allow communication and interoperability of different devices and networks; dynamic and massively distributed device networks; natural feeling human interfaces to support the intuitive use of the system; and dependability and security.

Although the concept of ambient intelligence may seem identical to that of ubiquitous computing, the main difference lies in the fact that ambient intelligence incorporates a strong component of Artificial Intelligence (AI), encompassing contributions from machine learning, multiagent platforms and robotics. For instance, ambient intelligence

may include works about voice technologies, artificial vision, natural language and knowledge management [Eisaku and Yasuhiru, 2006], so that they can show a more sensitive, responsive, adaptive and ubiquitous behavior.

In a more specific way, the existence of three areas of the AI contributing to the development of ambient intelligence [Cook et al., 2007] can be stated. The first area is **sensing**, which is required as ambient intelligence systems rely on information about the environment. The perception task can be accomplished using a variety of wireless sensors which are usually quite small. These sensors may optionally perform some local processing of the data before sending it to some servers which gather all the data from the different sensors to produce more accurate and complete information.

The second area is **acting**, which provides some means to the ambient intelligence system to execute some actions on the environment and affect its users. Some systems act through robots which implement some humanoid appearance and behavior, thus providing a human-like way of communication, while other systems rely on other output interfaces, such as panels.

Finally, a **reasoning** process sits between the sensing and acting tasks, in order to achieve the intelligent behavior of the system.

2.3 Context-Aware Systems

Ubiquitous computing and ambient intelligence introduce some challenges that are not present in traditional computing, since systems must now be capable of working in highly dynamic environments [Henricksen and Indulska, 2005]. For instance, in some mobile devices it may be useful that applications react to the rapidly changing context [Sánchez-Pi, 2011], such as the location, time, etc. For that reason, context-aware systems appear as a field of ubiquitous and pervasive computing which refers to systems that adapt their behavior to the current context. A more elaborated definition is provided in [Sánchez-Pi, 2011]:

> *"A system is context-aware if it can extract, interpret and use context information and adapt its functionality to the current context of use. The challenge for such systems lies in the complexity of capturing, representing and processing contextual data."*

The term *context-aware* applied to the field of computer science was first used in 1994 in [Bill N. Schilit and Want, 1994]. Two years before, in 1992, the active badge location system made its appearance [Want et al., 1992]. This system, which is considered to be the first context-aware one, serves for the location of people in an office environment. Those people were wearing badges which provided information of their location to a central location service.

To better understand the concept of context-aware systems, it is important to previously realize what is the meaning of *context* in the area of computer science. Although many definitions have been proposed for the term, one of the most accurate meanings is given by [Dey and Abowd, 1999]:

> *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."*

Moreover, Dey and Abowd present different categories of context types, which can be classified as primary and secondary. The primary context types are location, identity, time and activity; which answer the questions *where*, *who*, *when* and *what* respectively. The secondary context types contains other pieces of contextual information, such as details about places, people or objects (e.g., a person email address).

A simple approach to context-aware systems is known as *location-aware systems*, which is a subset of context-aware systems only dealing with location information. These systems are rapidly growing due to the spread of mobile devices and applications [Baldauf et al., 2007] and as a consequence of it, it can be found many systems and devices that provide some functionality based on their current position or proximity to other devices. Some examples of this kind of systems are navigators, smartphones, cameras with geotagging functionality, social networks, game consoles, etc. Many of these systems uses technologies such as Global Positioning System (GPS) and wireless networks for retrieving their current location.

2.4 Context Models

For a context-aware system to work, it is required that the contextual information is somehow formalized and stored. A context model formalizes, stores and shares contextual information in a way that can be processed by a computer, so that a context-aware system can use this knowledge to suuport its work [Baldauf et al., 2007].

According to [Strang and Linnhoff-Popien, 2004], most context-aware systems highly demand six main requirements on context models. First, they must have a distributed composition, as pervasive devices lack of a central system and the composition of a context model varies notably in terms of time, source and network topology. Secondly, it is desirable to be able to partially validate contextual knowledge, as the complexity of contextual interrelationships make the modeling intention error-prone. In the third place, given that the quality, richness and accuracy of the information varies over time depending on the technology used a context model should support quality and richness indication. Moreover, as contextual information uses to be incomplete or ambiguous mostly due to technology

limitations, the context model must be able to deal with this ambiguity. In the fifth place, contextual facts and interrelationships must be described in a precise and traceable manner, so that each participant in an ubiquitous interaction shares the same interpretation of the data exchanged and its semantics. Finally, a context model must be applicable within existing ubiquitous computing infrastructures.

2.4.1 Context Modeling Approaches

[Strang and Linnhoff-Popien, 2004] summarizes the most relevant context modeling approaches, based on the data structures for the representation and exchange of contextual information.

- **Key-Value** models: they are the simplest data structure for context modeling, and store the value of contextual information identified by keys. Although these models are easily managed, they lack of a more sophisticated structure.
- **Markup scheme** models: they model the contextual information in a hierarchical data structure consisting of markup tags with attributes and contents.
- **Graphical** models: there are several approaches, Unified Modeling Language (UML) diagrams are used to model contextual information. The use of UML is appropiate for context modeling due to its generic structure. Moreover, it is applicable to derive an ER model, which is useful to elaborate the structure of a relational database.
- **Object oriented** models: object orientation provides many powerful features for context modeling, such as encapsulation, reusability, inheritance, etc. Existing approaches use objects to represent context types and encapsulates the details. The context processing logic is provided through interfaces.
- **Logic based** models: these approaches use facts, expressions and rules to define a context model. An inference process derives new facts from the existing knowledge, composed of the existing facts and rules. The contextual information must be represented formally as facts. One of the earliest approaches was published by [McCarthy and Buvac, 1997].
- **Ontology based** models: ontologies are a very promising instrument for context modeling, as they can represent a description of concepts and relationships in a very expressive and intuitive way.

2.5 Natural Interaction

In 1950, Alan Turing wrote his article *Computing Machinery and Intelligence* [Turing, 1950], in which he described a variation of the *imitation game*, later known as

the *Turing test*. The main idea behind this test is that a human is able to decide whether he is having a conversation with a machine or with another human. For a machine to win this game, it must be able to interpret and understand the conversation and to generate appropiate human-like answers. The Turing test has been strongly influential over many subfields of Artificial Intelligence (AI) such as the Natural Language Processing (NLP). Half a century later, the concept of natural interaction was born.

A very important aspect of ubiquitous computing and ambient intelligence has to do with interaction. The progressive establishment of computers in the society requires new ways of interaction and, as many people with no technological training nor experience in interacting with computers gain access to these devices, simple and natural interfaces must be implemented. The goal, as it was previously stated, is that computers turn out to be invisible for humans; thus requiring the achievement of new forms of interaction in which the user communicates with a computer as if he were interacting with other humans. Additionally, this way of interaction would also increase the accesibility of the system to disabled users.

2.5.1 Implicit Interaction

In the first place, it is important to understand the concept of implicit interaction [Schmidt, 2000]. When humans interact with themselves, a lot of information is exchanged implicitly through the participants behavior, gestures, the context, etc.; which supports, complements or disambiguates the information explicitly exchanged. Nowadays, when a user interacts with a computer, he mostly inputs some commands in an explicit form through a Command Line Interface (CLI), a Graphical User Interface (GUI) or through the speech. On the other hand, implicit HCI does not require from the user explicit instructions or commands; instead, the system understands an action carried out by the user as input. [Schmidt, 2000] provides the next definition:

> *"Implicit human computer interaction is an action, performed by the user that is not primarily aimed to interact with a computerized system but which such a system understands as input."*

To attain such approach, the system must be able to infer what the user is seeking with his behavior, and for this reason it may need some understanding of the context and how the user can behave in it; thus requiring some mechanisms for both perception of the environment and interpretation of which is perceived. The fact that implicit interaction does not require from the user to explicitly emit commands to the computer establishes the link between this concept and ambient intelligence (which provides some response to user actions proactively) and consequently with context-aware systems.

2.5.2 Natural Language Processing

The Natural Language Processing (NLP) is a field of computer science which groups a set of computational techniques for dealing with natural language, with the main purpose of achieving human-like language processing capabilities. For this reason, NLP is often considered as a discipline of AI, and its problems are usually allocated as AI-hard problems. Moreover, this field is also strongly influenced by other disciplines, such as linguistics, which focuses on formal models of language; and cognitive psychology [Liddy, 2003].

Starting in the late 1940s, the research on NLP during the first years was mostly focused on machine translation. In the beginnings, machine translation was basically performed by direct word-by-word translation and reordering, thus leading to very poor results. In 1957 Chomsky introduced the idea of generative grammars [Chomsky, 1957], resulting in one of the earlier contributions from linguistics to NLP. In those years, some other areas of NLP emerge, such as speech recognition and synthesis.

Another important progress in the field of NLP was achieved in the late 1980s, when machine learning algorithms started to be applied for language processing. Previously, most NLP systems were based on complex rules, which were manually coded; and the application of machine learning and statistical models (instead of pure linguistical ones) produced more reliable systems.

In the last years, the field has been growing rapidly mostly due to the increase of the computing power and the availability of large amounts of data in the Internet [Liddy, 2003]. Some of the tasks of NLP includes natural language recognition, understanding and generation, Optical Character Recognition (OCR) and speech recognition, machine translation, discourse analysis, sentiment analysis, question answering, machine translation and many others.

2.5.3 Natural User Interfaces

In 1980, Richard Bolt described a computer system in which the user could place some shapes in certain places of a large display by using both voice commands and pointing gestures [Bolt, 1980]. Despite the simplicity of this system, it pointed out a new way of human-computer interaction, which did not require the use of computer input devices such as a keyboard or a mouse. This system was one of the earliest ones to include what was later known as a Natural User Interface (NUI).

A NUI can be defined as a computer interface in which the interaction is performed without requiring specific input devices. These kind of interfaces follow the classical CLIs, where the commands were required to be typed in the screen; and GUIs, where the user interacts with the computer through images which usually served as metaphors for daily objects.

A key concept related to NUIs has to do with multimodality. In a typical human-

human conversation, each individual does not only communicate through spoken language, but also body language takes an important role. Multimodal interfaces allow the user to communicate through different modalities, such as speech, written language, gestures, etc; leading to a more natural and transparent interactive experience.

The use of multimodal NUIs provides several advantages [Oviatt and Cohen, 2000]. In the first place, they have the potential for increasing the system accesibility, usability and comfortability. These systems usually have a very simple learning curve, and can often be used by users of different ages and skills, and even by impaired users. Moreover, multimodal interfaces usually allow the user to switch between different modalities to adapt to the user necessities. Secondly, multimodal interfaces usually improve the robustness and accuracy of the system, a property that can be achieved by the mutual disambiguation of the different inputs. Finally, these interfaces provide great expressive power to deal with the exchange of information, given that each interface complements the other ones.

While most of the natural interfaces nowadays mainly serve as a supplement for standard GUIs, in the future those interfaces will eventually replace them, as well as current GUIs have replaced the already obsolete CLIs.

2.5.4 Natural Interaction

The previous concepts will serve as the proper basis to define what natural interaction is. According to [Bernsen, 2000], *natural interaction* is the interaction between humans and computer systems which takes place in the ways in which humans normally exchange information with one another. Based on this definition, it can be realized that natural interaction is the kind of interaction used with a NUI.

As previously stated, an important advantage of these interaction techniques is that the user does not require any technical knowledge or ability to deal with the system, but only the capability of interacting with other humans. It involves that these kind of systems must be able both to interpret and understand what the user is expressing as well as to ellaborate its own interventions in a way that can be understood by the user.

A natural interaction system combines techniques from very different fields. Research in natural interaction not only is strongly influenced by areas from the computer science, such as the human-computer interaction, the knowledge engineering or the natural language processing; but also from other human sciences such as the psycholinguistics and the sociolinguistics.

A key issue for a natural interaction system is that knowledge supporting the interaction has different natures. Besides the purely interactive, the system may require additional types of knowledge such as the operational (which has to do with tasks), the circumstantial, the emotional, knowledge about the user or even knowledge about its own status and goals. The management of some of this knowledge may be highly complex and computationally expensive. However, removing some of this knowledge will reduce the naturality of the

system, leading to a more mechanical behavior for the aspect regarding the discarded knowledge [Calle et al., 2008].

2.6 Situation Models

As stated in chapter 1, an important component of a natural interaction system has to do with the management of circumstantial knowledge, i.e., the context of the interaction. The term *situation* was used by [Connolly, 2001] to refer to the non-linguistic context. It has also been used in previous projects based in the cognitive architecture proposed in [Calle, 2004]. Throughout this work, term *situation model* will be used to refer to a context model which stores and processes the circumstantial knowledge.

A situation model turns out to be a key component in those cases where the context is not fixed for the interaction. The advantages of the inclusion of a situation model in a natural interaction system have been listed in section 1.1.

2.7 Applications

The purpose of this section is to discuss some natural interaction systems which have either contributed to the research in situation models, or where these models have been successfully applied.

2.7.1 TRIPS

TRIPS (The Rochester Interactive Planning System) [Ferguson and Allen, 1998] is a system which integrates several AI techniques in order to provide the user with an interactive problem-solving assistant in a logistic domain. This research project was carried out in the University of Rochester's Department of Computer Science in the late 1990s.

TRIPS is built over the TRAINS system [Allen et al., 1994], although it allows planning on more complicated domains. The main purpose of TRIPS system is to assist the user to build plans in crisis situations. The generated plan may have to meet some specific constraints such as time, cost or weather conditions. Thus, the system may require some circumstantial knowledge to generate an adequate plan.

The TRIPS architecture (shown in figure 2.1) is divided in three groups. The first one has to do with modality processing, which mostly means input recognition and output generation and the third group contains the specialized reasoners such as the planner or the scheduler, each one being able to solve some particular problems. Meanwhile, the second

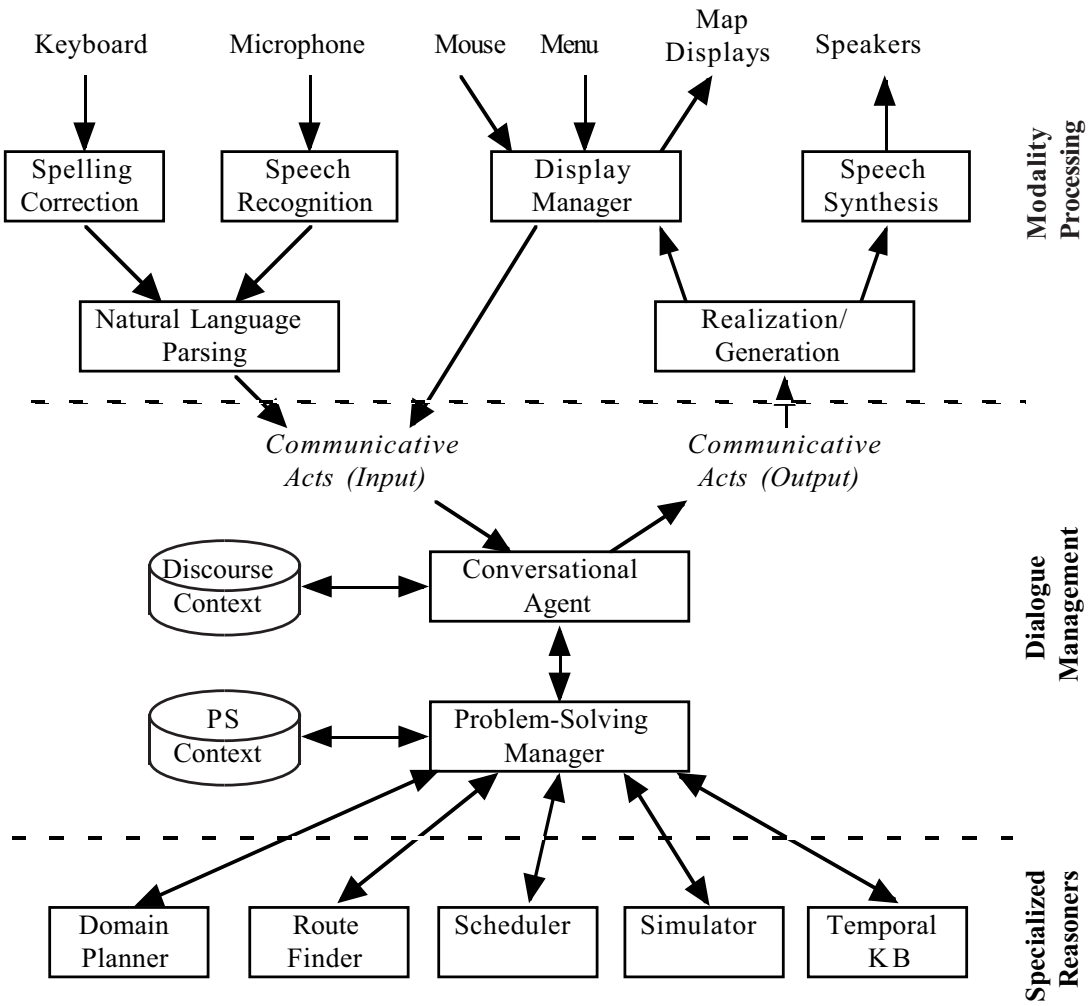


Figure 2.1: TRIPS architecture

group communicates the previous two by managing the conversation and coordinating the reasoners.

TRIPS project-solving manager is a component within the dialogue management group, which deals with a database which stores the problem-solving context and with specialized reasoners, and whose purpose is to keep a representation of the task to be performed, coordinate the generation of a feasible and coherent plan and maintain the state of the solution.

The other key component in TRIPS dialogue management is the conversational agent. This component is in charge of dealing with the problem-solver manager as it interacts with the user. For this interaction to be performed, the conversational agent considers the input by means of communicative acts, which are interpreted along with the discourse context to discover what the user means. In coordination with the problem-solver manager, it generates an output which can be expressed again as communicative acts.

A new core architecture for TRIPS was proposed in 2001 [James Allen and Stent, 2001] in order to emphasize the distinction between domain planning and discourse planning, thus increasing the portability and maintainability of the system.

More information on TRIPS system, its domains and some demonstration movies can be found in `http://www.cs.rochester.edu/research/cisd/projects/trips`. Additionally, more information of its preceeding project TRAINS can be found in `http://www.cs.rochester.edu/research/cisd/projects/trains`.

2.7.2 Deep Map

Deep Map [Malaka and Zipf, 2000] is a research framework of a tourist information system whose development took place in the early 2000s. The purpose of Deep Map is to build a mobile solution which serves as a trip planner and as a city navigator. The system would take in consideration the user interests and needs, the social and cultural background and some other contextual information such as the weather conditions or traffic. While the system would provide complex functionality and combine several technologies, it was thought to still be accessible for untrained users.

The core component for Deep Map is the geographical information system (GIS), which manages the location information and relates it to the user needs. Additionally, as the user may require some historical information about any particular location, the architecture also needs some temporal capabilities in its databases.

For that reason, Deep Map contains a 4D-database which represents spatial information in three dimensions as well as temporal or historical information. The third dimension is required to store knowledge about the visibility of objects from a given location, while the fourth dimension is required to store temporal evolution of such objects. Regarding the interaction with the system, it requires to be intuitive and usable and, as it is intended to be used in a situation where the user may be walking or driving, it also has to be unintrusive. Therefore, the interaction by means of natural language (both for the input and the output) is clearly adequate. Actually, Deep Map presents different interfaces, such as a classical GUI, 3D models, natural language, etc.

2.7.3 SmartKom

The importance of multimodality in computer interfaces and the advantages it provides for the interaction between humans and computers have been pointed out in previous sections. The SmartKom system aims for an access to its functionalities which is efficient and intuitive for the user by providing symmetric multimodality [Wahlster, 2006], that is, not only understanding a multimodal input, but also generating its own multimodal output using the same modalities that those considered for the input. SmartKom interaction management is based on models to represent and reason about a certain type of knowledge,

such as user model, task model, domain model, etc. One of the major scientific goals of SmartKom was to understand imprecise or ambiguous input through the fussion of diverse inputs for the different modalities.

SmartKom was designed as a four-years project. It lasted from September, 1999 to September, 2003 and 25.7 million euros were allocated to it. This funding mainly came from the German Federal Ministry of Education and Research (BMBF) and from other private partners [Blocher, 2006].

As SmartKom is motivated for non-desktop scenarios, the situational context takes an important role of the interaction process. For this reason, the SmartKom system implements a context model to manage the circumstantial knowledge. The context model is highlighted in figure 2.2, which shows the general architecture for the SmartKom system [Herzog and Reithinger, 2006], following the component-and-connector viewtype defined in [Clements et al., 2003].

The context model for the SmartKom system is described in [Porzel et al., 2006]. This model contemplates four different types of context: the dialogical context (the knowledge on what has been said), the ontological or domain context (the knowledge of concepts on the domain), the user context (the knowledge about the interlocutors) and the situational context (the material aspect of the context, such as the time or the place).

One of the main ideas of SmartKom is that the kernel system can be used in different scenarios (manifestations of the system, such as home, office, mobile, etc.), modalities (such as speech, gesture, mimics or affectives) and domains (topics of conversation, such as train schedules, cinema timetables or hotel bookings). As the scenario information and the domain knowledge are closely related, SmartKom integrates the situational and the domain knowledges together in what has been called the *modeler knowledge* module.

This module performs what is known as the decontextualization process, which builds a context-free statement based on a context-dependent statement and on contextual information. This process receives dynamic spatio-temporal information and a set of intention hypothesis based on the user utterances as input. Then, it decides what situational and domain information is relevant for resolving the contextual reference and computes scores for each different hypothesis based on its contextual coherence. Finally, the process returns a decontextualized intention hypothesis.

More information about the SmartKom project, as well as some demonstration videos can be found in `http://www.smartkom.org/start_en.html`.

2.7.4 Apple Siri

Siri is a personal assistant application which was introduced by Apple in the iPhone 4S smartphone in 2011. Siri implements speech recognition, and the user can communicate with it in a natural way in order to ask for information, to ask for an action to be performed

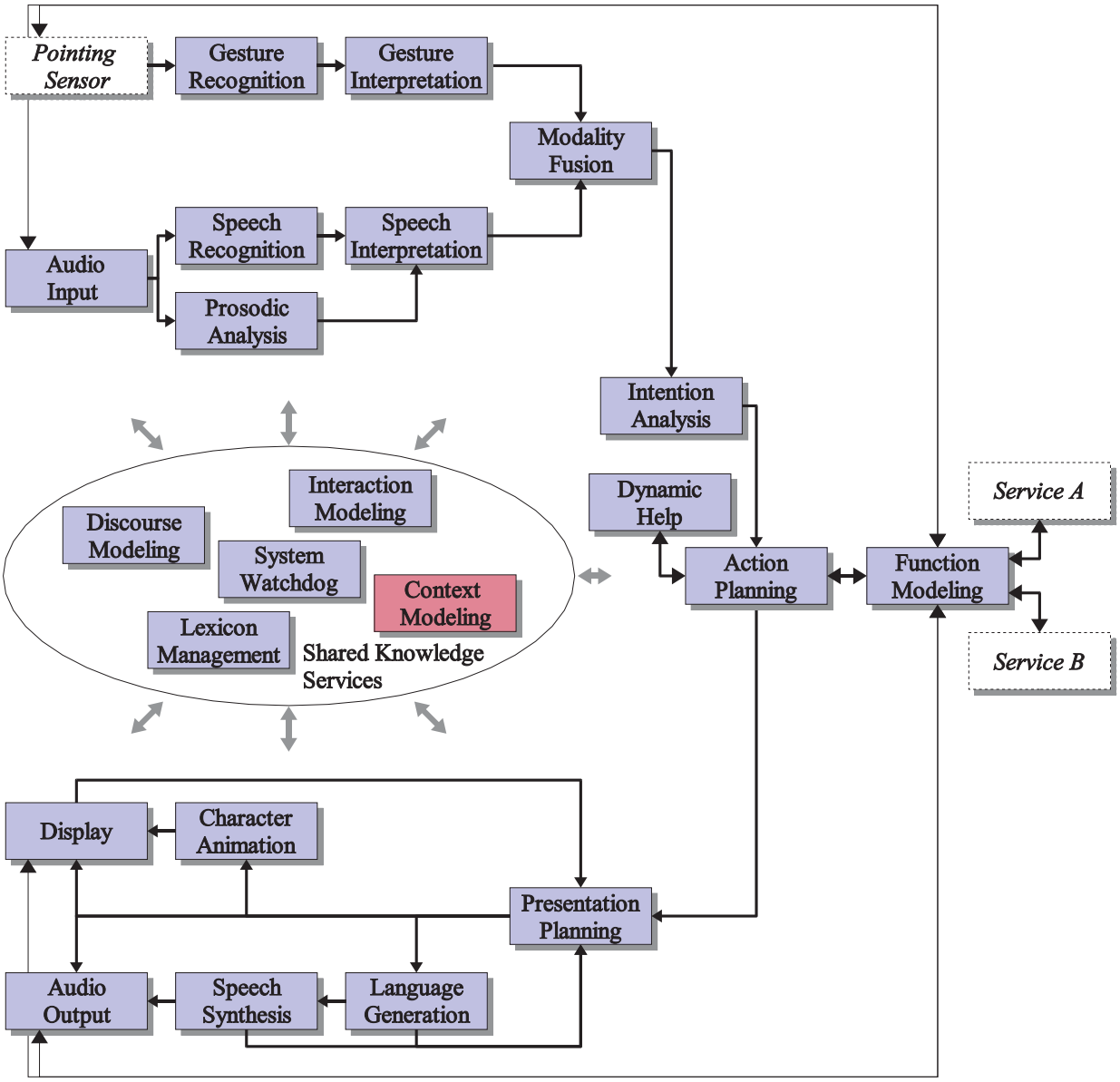


Figure 2.2: SmartKom architecture

or just for entertaining.

Siri takes advantage of the iPhone localization mechanisms to know the user location, and online information can provide much more details about the material aspect of the context in that place, such as weather or traffic information. This situational knowledge enables Siri to provide some context-aware functionalities such as looking for places near your location and starting a navigation to that place, creating new reminders or alarms at a given time or asking for the weather at a certain place. The assistant may refer to additional information in order to increase the naturality of the interaction, for example, the user can ask Siri to create a reminder when he leaves work; which requires previous

knowledge on the user schedule.

A reference to Siri as a human-like interaction system is included in the Siri FAQs [Apple.com, 2012] in the Apple website:

> *"You can speak to Siri as you would to a person - in a natural voice with conversational tone. If you want to know what the weather will be like tomorrow, simply say "What will the weather be like tomorrow?" Or "Does it look like rain tomorrow?" Or even "Will I need an umbrella tomorrow?" No matter how you ask, Siri will tell you the forecast."*

More information about Siri can be found in the next website: `http://www.apple.com/iphone/features/siri.html`

2.8 Conclusions

The concept of ubiquitous computing presents a future where computers not only are located everywhere, but also the interaction with them vanishes in the background, allowing humans to focus on the topic of the communication, instead of the communication itself.

When humans talk each other in a daily context, they can put all their efforts on *what* they are saying, not paying attention to *how* they are moving their lips or how they are gesturing. Unfortunately, current computer interfaces are not really contributing for the achievement of this ubiquity, as people still have to focus on whether they pressed the correct key or how they are moving the mouse.

Natural User Interfaces (NUIs) are a step forward in achieving that goal, as far as they provide a way of interacting with the system which resembles human-human interaction. Two main features of these interfaces are multimodality and context-awareness. The first one involves using more than one way of interaction in the communication process, whereas the second allows a more realistic dialogue which takes into account the context of the interaction.

The purpose of this project, as it was stated in section 1.2, is to develop a tool to manage circumstantial knowledge, thus providing context-awareness capabilites to a natural interaction system.

This page has been intentionally left blank.

Chapter 3

Project Description

The main purpose of this chapter is to describe the project by obtaining and specifying the requirements for the software product (sections 3.1 and 3.2), which will provide enough information to be able of generating an analysis and a detailed system design in a further stage (see chapter 4) which satisfies those requirements.

Sections 3.3 and 3.4 include a review of the previous existing work and a feasibility study respectively, to conclude whether the system can be developed and which environmental constraints must be satisfied for this to happen.

Section 3.5 defines a set of tests to be performed after the software is developed, to validate the software against the specified requirements. Section 3.6 describes the development methodology and lifecycle which will be followed for the current project. Finally, section 3.7 documents the project planning, both in terms of schedule and costs.

3.1 Requirements Elicitation

In order to perform a requirements specification, a previous work on requirements elicitation must be carried out. The aim of this process is that the requirements engineer gathers from the users or customers the requirements of a system. For this project, the tutor will play the role of the customer, while the student will act as a requirements engineer, analyst, designer and programmer.

An introductory meeting was carried out the first week of December, 2011. This meeting took the format of an open interview between the customer and the engineer, where the problem of how to improve a natural interaction system in order to manage situational knowledge was discussed.

Later that month, a second meeting was arranged, also with the format of an open interview, to discuss in more detail the theoretical background of the project, as well as

some existing applications of situation models in natural interaction systems.

A semi-structured interview took place in February with the purpose of settling the report structure and to choose the most appropiate development methodology for the project (section 3.6).

Additionally, three semi-structured interviews were carried out for the beginning of each development cycle (in March, April and May), to obtain, specify and update the software requirements. In the last two interviews, the user could refine the software requirements given its own experience with the generated prototypes from the previous cycles.

3.2 Software Requirements Specification

For the requirements specifications task, the IEEE recommended practices [IEEE, 1998] will be followed. According to these guidelines, a good specification must address the software functionality, the external interfaces, performance issues, other non-functional features and design or implementation constraints. Moreover, the requirements specification should be:

- **Correct**: every requirement is one that the software shall meet according to the user needs.
- **Unambiguous**: every requirement has one single interpretation.
- **Complete**: the document reflects all significant software requirements.
- **Consistent**: requirements must not generate any conflict with each other.
- **Ranked for importance and/or stability**: every requirement must indicate either its importance or its stability.
- **Verifiable**: every requirement must be verifiable, that is, there exists some process to verify that the software complies with every single requirement.
- **Modifiable**: the structure of the specification allows performing changes to the requirements in a simple, complete and consistent way.
- **Traceable**: the origin for every requirement is clear and it can be easily referenced in further stages.

Table 3.1 provides a template for requirements specification.

ID	Requirement ID.
Name	Requirement name.
Description	Requirement description.
Cycle	The development cycle in which the requirement was defined.
Priority	Requirement priority according to its importance. A requirement can be identified either as *essential*, *conditional* or *optional*.
Stability	Requirement stability, defined either as *stable* or *unstable*.

Table 3.1: Template for software requirements specification

3.2.1 Functional Requirements

This subsection describes the user requirements related to the system functionality.

ID	FR-01.				
Name	Situation model selection.				
Description	The system shall allow the user to select a situation model in order to work on it.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.2: Requirement FR-01 (Situation model selection)

ID	FR-02.				
Name	Situation model management.				
Description	The system shall provide means to manage the situation model knowledge.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.3: Requirement FR-02 (Situation model management)

ID	FR-03.				
Name	Situation model simulation.				
Description	The system shall include mechanisms to perform a simulation over the situation model.				
Cycle	First.	**Priority**	Conditional.	**Stability**	Stable.

Table 3.4: Requirement FR-03 (Situation model simulation)

ID	FR-04.				
Name	Persistent storage.				
Description	The system shall store all the model knowledge persistently.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.5: Requirement FR-04 (Persistent storage)

3.2.1.1 Management

This subsection describes in further detail the user requirements for the network management functionality.

ID	FR-MA-01.				
Name	Networks management.				
Description	The system shall provide means to the user to create and delete networks.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.6: Requirement FR-MA-01 (Networks management)

ID	FR-MA-02.				
Name	Networks selection.				
Description	The system shall provide means to the user to select a network in order to work with it.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.7: Requirement FR-MA-02 (Network selection)

ID	FR-MA-03.				
Name	Networks properties.				
Description	Networks are identified by a name and can have an optional description.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.8: Requirement FR-MA-03 (Networks properties)

ID	FR-MA-04.				
Name	Network items.				
Description	Networks may contain nodes and links between nodes.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.9: Requirement FR-MA-04 (Network items)

ID	FR-MA-05.				
Name	Network items properties.				
Description	Networks items are identified by a numeric identifier and may have an optional description.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.10: Requirement FR-MA-05 (Network items properties)

ID	FR-MA-06.				
Name	Network items activeness.				
Description	Networks items may be active or inactive.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.11: Requirement FR-MA-06 (Network items activeness)

ID	FR-MA-07				
Name	Network items hierarchy.				
Description	The system shall support a future extension to structure the network items in a hierarchy.				
Cycle	First.	**Priority**	Conditional.	**Stability**	Unstable.

Table 3.12: Requirement FR-MA-07 (Network items hierarchy)

ID	FR-MA-08.				
Name	Network items cost.				
Description	Network items have an associated basic cost.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.13: Requirement FR-MA-08 (Network items cost)

ID	FR-MA-09.				
Name	Network items coordinates.				
Description	Networks items are located in a geometric coordinates system.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.14: Requirement FR-MA-09 (Network items coordinates)

ID	FR-MA-10.				
Name	Features.				
Description	The system shall provide means to create and delete user-defined features.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.15: Requirement FR-MA-10 (Features)

ID	FR-MA-11.				
Name	Features properties.				
Description	Features are identified by their name.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.16: Requirement FR-MA-11 (Features properties)

ID	FR-MA-12.				
Name	Network items features.				
Description	The system shall provide means to the user to assign features to the network items.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.17: Requirement FR-MA-12 (Network items features)

ID	FR-MA-13.				
Name	Network planes management.				
Description	The system should provide means for the user to create and delete planes for a previously selected network.				
Cycle	First.	**Priority**	Optional.	**Stability**	Stable.

Table 3.18: Requirement FR-MA-13 (Network planes)

ID	FR-MA-14.				
Name	Network planes properties.				
Description	Network planes are identified by their name, and must have an associated background picture.				
Cycle	First.	**Priority**	Optional.	**Stability**	Unstable.

Table 3.19: Requirement FR-MA-14 (Network planes properties)

ID	FR-MA-15.				
Name	Network planes selection.				
Description	The system shall provide means to the user to select a plane within the selected previously selected network.				
Cycle	First.	**Priority**	Optional.	**Stability**	Stable.

Table 3.20: Requirement FR-MA-15 (Network planes selection)

ID	FR-MA-16.				
Name	Nodes management.				
Description	The system shall provide means to the user to create, edit and delete nodes within a previously selected network and plane.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.21: Requirement FR-MA-16 (Nodes management)

ID	FR-MA-17.				
Name	Nodes creation.				
Description	The user shall be able to create nodes with a graphic interface by placing them over the background image of the selected plane.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.22: Requirement FR-MA-17 (Nodes creation)

ID	FR-MA-18.				
Name	Nodes edition.				
Description	The user shall be able to select a node and edit its properties.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.23: Requirement FR-MA-18 (Nodes edition)

ID	FR-MA-19.				
Name	Nodes properties.				
Description	Besides the properties common to all network items (specified in requirements FR-MA-08 to FR-MA-13), a node may have an action radius which would be dependent on the localization technology.				
Cycle	First.	**Priority**	Conditional.	**Stability**	Unstable.

Table 3.24: Requirement FR-MA-19 (Nodes properties)

ID	FR-MA-20.				
Name	Links management.				
Description	The system shall provide means to the user to create, edit and delete links within a previously selected network and plane.				
Cycle	First.	**Priority**	Conditional.	**Stability**	Unstable.

Table 3.25: Requirement FR-MA-20 (Links management)

ID	FR-MA-21.				
Name	Links creation.				
Description	The user shall be able to create links with a graphic interface by choosing the start and end nodes of the link.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.26: Requirement FR-MA-21 (Links creation)

ID	FR-MA-22.				
Name	Links properties.				
Description	Besides the properties common to all network items (specified in requirements FR-MA-08 to FR-MA-13), a link may be bidirected or not.				
Cycle	First.	**Priority**	Conditional.	**Stability**	Unstable.

Table 3.27: Requirement FR-MA-22 (Links properties)

ID	FR-MA-23.				
Name	Edition interface zoom.				
Description	The user should be able to zoom in or zoom out the plane background image and the network items placed over it to obtain a higher precision in the edition process.				
Cycle	First.	**Priority**	Optional.	**Stability**	Stable.

Table 3.28: Requirement FR-MA-23 (Edition interface zoom)

ID	FR-MA-24.				
Name	Situational knowledge taxonomy.				
Description	The system shall classify the situational knowledge in seven different aspects: *spatial*, *temporal*, *ambiental*, *political*, *sociocultural*, *operational* and *semiotical*.				
Cycle	Third.	**Priority**	Essential.	**Stability**	Stable.

Table 3.29: Requirement FR-MA-24 (Situational knowledge taxonomy)

ID	FR-MA-25.				
Name	Situations management.				
Description	The system shall provide means to the user to create and delete situations within the situational aspects specified in FR-MA-01.				
Cycle	Third.	**Priority**	Essential.	**Stability**	Stable.

Table 3.30: Requirement FR-MA-25 (Situations management)

ID	FR-MA-26.				
Name	Situations properties.				
Description	Situations are identified by a name and can have an optional description.				
Cycle	Third.	**Priority**	Essential.	**Stability**	Stable.

Table 3.31: Requirement FR-MA-26 (Situations properties)

ID	FR-MA-27.				
Name	Network items cost factors.				
Description	The system shall provide means to the user to assign cost factors (cost multipliers) to network items for each defined situation.				
Cycle	Third.	**Priority**	Conditional.	**Stability**	Stable.

Table 3.32: Requirement FR-MA-27 (Network items cost factors)

ID	FR-MA-28.				
Name	Features cost factors.				
Description	The system shall provide means to the user to assign cost factors to defined features for each situation.				
Cycle	Third.	**Priority**	Essential.	**Stability**	Stable.

Table 3.33: Requirement FR-MA-28 (Features cost factors)

3.2.1.2 Simulation

This subsection describes in further detail the user requirements for the simulation functionality.

ID	FR-SI-01.				
Name	Network selection				
Description	The system shall provide means to the user to select a network in order to perform a simulation on it.				
Cycle	Second.	**Priority**	Essential.	**Stability**	Stable.

Table 3.34: Requirement FR-SI-01 (Network selection)

ID	FR-SI-02.				
Name	Network display				
Description	When a network is selected, the system shall display the network items and the network background image.				
Cycle	Second.	**Priority**	Essential.	**Stability**	Stable.

Table 3.35: Requirement FR-SI-02 (Network display)

ID	FR-SI-03.				
Name	Person placement				
Description	If the user clicks on the network, the system shall place a person on the network, over the location that the user has clicked.				
Cycle	Second.	**Priority**	Essential.	**Stability**	Stable.

Table 3.36: Requirement FR-SI-03 (Person placement)

ID	FR-SI-04.				
Name	Person motion				
Description	Once the person is placed over the network, the system shall allow its motion using the keyboard direction arrows and the mouse.				
Cycle	Second.	**Priority**	Essential.	**Stability**	Stable.

Table 3.37: Requirement FR-SI-04 (Person motion)

ID	FR-SI-05.				
Name	Physical layer simulation				
Description	Additionally, the system should provide some simulators for a physical layer for localization, such as GPS or Radio Frequency IDentification (RFID), which will move the person across the network.				
Cycle	Second.	**Priority**	Conditional.	**Stability**	Stable.

Table 3.38: Requirement FR-SI-05 (Physical layer simulation)

ID	FR-SI-06.				
Name	Simulation services				
Description	The simulation shall provide the next services: description, route and navigation.				
Cycle	Second.	**Priority**	Essential.	**Stability**	Stable.

Table 3.39: Requirement FR-SI-06 (Simulation services)

ID	FR-SI-07.				
Name	Description				
Description	A service for a context description, based on the descriptions of the network items which are closest to the person, shall be available as an eventual service.				
Cycle	Second.	**Priority**	Essential.	**Stability**	Stable.

Table 3.40: Requirement FR-SI-07 (Description)

ID	FR-SI-08.				
Name	Routing				
Description	A service for describing a route between two points shall be available as an eventual service.				
Cycle	Second.	**Priority**	Essential.	**Stability**	Stable.

Table 3.41: Requirement FR-SI-08 (Routing)

ID	FR-SI-09.				
Name	Navigation				
Description	A navigation service shall be available, which describes the steps for achieving a goal as the person moves through the network.				
Cycle	Second.	**Priority**	Essential.	**Stability**	Stable.

Table 3.42: Requirement FR-SI-09 (Navigation)

ID	FR-SI-10.				
Name	Source and goal location				
Description	The source location will be based on the person location, while the goal location will be selected by the user in the network.				
Cycle	Second.	**Priority**	Essential.	**Stability**	Stable.

Table 3.43: Requirement FR-SI-10 (Source and goal location)

ID	FR-SI-11.				
Name	Minimum cost path				
Description	For the routing and navigation services, the path to the goal will have minimum cost.				
Cycle	Second.	**Priority**	Essential.	**Stability**	Stable.

Table 3.44: Requirement FR-SI-11 (Minimum cost path)

ID	FR-SI-12.				
Name	Situations selection				
Description	The system shall provide means to the user to select several user-defined situations which will apply during the simulation.				
Cycle	Third.	**Priority**	Essential.	**Stability**	Stable.

Table 3.45: Requirement FR-SI-12 (Situations selection)

3.2.2 Non-Functional Requirements

This subsection describes the user requirements which are not related directly to what the system shall do, but instead how the system shall be, that is, describes system features rather than system functionality.

3.2.2.1 Scalability Requirements

This subsection describes the user requirements referring to how the application shall scale when the size of the knowledge increases.

ID	NFR-SC-01.				
Name	Big networks.				
Description	The situation model and the management and simulation tool must be able to support big networks.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.46: Requirement NFR-SC-01 (Big networks)

3.2.2.2 Safety Requirements

This subsection describes the user requirements referring to how the application shall restrict access to some features only to authorized users.

ID	NFR-SA-01.				
Name	Authorized access.				
Description	The system shall ask the user for authentication information (username and password) before he can access to the model and select a network.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.47: Requirement NFR-SA-01 (Authorized access)

3.2.2.3 Interoperability Requirements

This subsection describes the user requirements referring to how the application shall behave over different platforms.

ID	NFR-IO-01.				
Name	Cross-platform.				
Description	The system shall operate in Microsoft Windows XP or higher, and Linux systems, either of 32 or 64 bits.				
Cycle	First.	**Priority**	Essential.	**Stability**	Stable.

Table 3.48: Requirement NFR-IO-01 (Cross-platform)

3.3 Previous Work

Some previous work on the database storing the situational information already exists. Figure 3.1 shows the architecture for the previously existing situational database, as described in [Rivero et al., 2007].

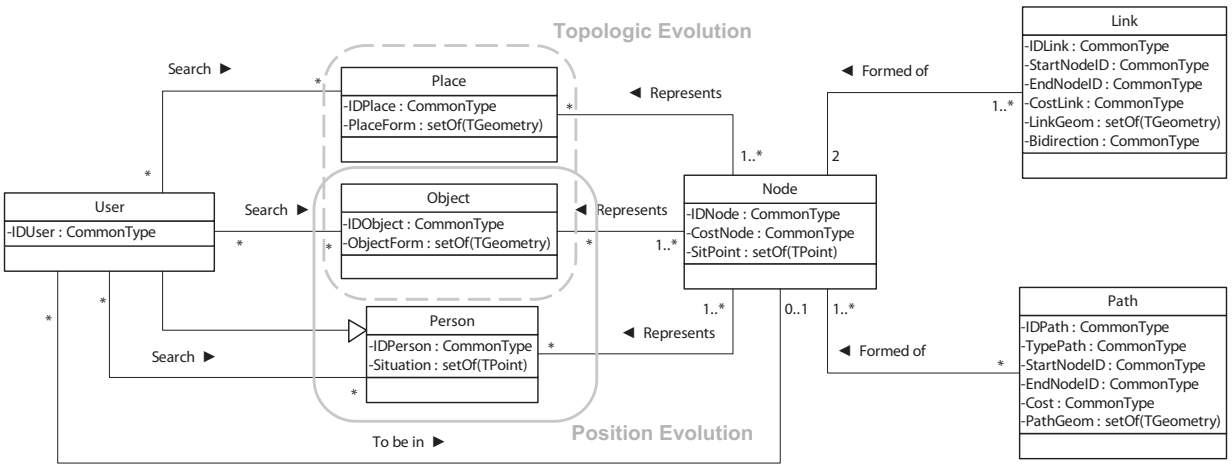


Figure 3.1: Architecture for the previous situation model database

For the purpose of this project, several modifications will be performed over this design. First, while the cost for each node or each link is currently stored within the item table, for this project, costs will be stores in an external relation, in order to increase the scalability with regard to the number of different situations having an associated cost. The creation of views may be required to take advantage of the database spatial capabilities.

Additionally, a characterization may be added to the network items (nodes and links), so that in the future some rule firing mechanism can be implemented over this characterization

to increase the performance and scalability of the system (e.g. assigning cost factors to items features instead of doing so for items themselves).

Finally, the database has to include some additional relations to support not only the model knowledge but also the edition knowledge, in order to be managed by the tool.

3.4 Feasibility Study

The purpose of this section is to discuss the project feasibility given the requirements specified by the user as well as technical and environmental constraints.

3.4.1 Technical Constraints

The application will follow a client-server model, thus the client and the server technical specifications for the application to work correctly have to be described separately.

3.4.1.1 Client Requirements

The client will execute the edition and the simulation software, which does not require too much computing power given that expensive operations, such as optimal paths calculations are intended to be performed in the server. However, some technical specifications are recommended for the user to obtain the best experience from the application:

- **Operating system:** Microsoft Windows XP, Windows Vista or Windows 7.
- **Processor:** Intel Core 2 Duo @2GHz and higher, or AMD equivalent.
- **RAM Memory:** 1024 MB.
- **Storage:** 2 GB of free space in HDD.
- **Network:** an Internet connection must be available.
- **Screen:** screen resolution of 1280×1024 or 1440×900 in widescreen mode.
- **Software:** Java Virtual Machine 1.6 or higher.

The use of Java Virtual Machine is justified as it provides cross-platform capabilities, as it was stated in requirement NFR-IO-01. Additionally, the development team is already trained in the development of applications and GUIs in the Java programming language.

As most commercial personal computers exceed by far these recommended requirements, no technical constraint in the client side compromises the project feasibility.

3.4.1.2 Server Requirements

As most of the application computations fall on the server side, the server must fulfill some specific technical requirements. Particularly, the server will run an Oracle Database to store all the model knowledge in a persistent way. Moreover, Oracle Spatial will load the network in main memory to run graph algorithms over it, which may require more RAM memory to be available.

The actual server specifications will depend on how the application will be used, being some key factors the size of the knowledge to be stored and the complexity of the networks. The next requirements should suffice for a common use of the application:

- **Operating system:** Microsoft Windows Server 2003, Windows Server 2008 or Windows Server 2011.
- **Processor:** AMD Opteron Series 2300 (Quad Core) @2GHz or higher, or Intel Xeon Series X3200 @2GHz or higher.
- **RAM Memory:** at least 4096 MB.
- **Storage:** HDD of 300GB at 15000 rpm.
- **Network:** an Internet connection must be available.
- **Software:** Oracle Database 11g Enterprise Edition.

The use of Oracle Database for persistent storage is justified for several reasons. First, the Advanced Databases Group already owns software licenses for Oracle Database, and the developers are trained in the use of this database management system. Moreover, the previous work described in section 3.3 was implemented over Oracle Database, thus the use of this software would allow code reusability and provides a better intuituion that the current project is feasible.

The Advanced Databases Group provides a server which fulfills these requirements for this project, so there are no technical constraints on the server side that could threaten the project feasibility.

3.4.2 Environmental Constraints

Environmental constraints will be classified attending to their nature. Thus, it can be found constraints that have to do with the application social environment and the project stakeholders (sociocultural constraints), others which are founded over economic aspects (economic constraints) and those who are related to the legal framework of the project (legal constraints). Finally, ethical considerations are discussed in appendix B.

3.4.2.1 Sociocultural Constraints

The edition tool is aimed at members of the academic community. The users of the application may not be familiar with this kind of advanced edition tools, and for that reason the application developed in this project must have an easy-to-use friendly interface. This fact is not an obstacle as many modern programming languages allow the creation of attractive user interfaces.

3.4.2.2 Economic Constraints

The work in this project could eventually be included in a research project, which may provide financial support for the research and development of a situation model and its associated management tools.

However, in the short term, the situation model developed for this project could be integrated with any other application in order to increase its funcionality. As smartphones and other intelligent handheld devices are rapidly gaining in importance, a situation model could provide a new way of interaction from which the user could take an advantage; as now the interaction would depend not only on what the user asks, but also on what the context is.

3.4.2.3 Legal Constraints

While the management and simulation application neither stores nor transmits confidential information about the users so far, future applications which work over the situation model will communicate sensible information such as the user profile and his location over time.

Many countries require that the transmission of this information is performed in such a way that third parties cannot gain access to this information, for example, by encrypting the transmitted information. In Spain, this requirement is specified in the article 104 of the *RD 1720/2007* [BOE, 2008]. Therefore, security requirements must be taken into account for further developments.

3.5 Validation Testing

The validation process aims to check that the development of the system results in a product that meets the requirements specified by the user, which are defined in section 3.2. For this reason, validation is an important task for guaranteeing the quality of the software product, and thus a validation plan containing complete and correct tests must be carefully designed.

The aim of this section is to describe the validation testing plan, which is composed by a set of validation tests, each of them covering at least one, but probably more than one user requirement. It can also happen that a requirement is covered by more than one validation test. In this case, all the tests covering the requirements must be successful in order to consider that the requirement is conveniently validated.

Each validation test is shown in a table containing a unique identifier, a name and a description. The test is defined by means of preconditions that the system must fulfill before the test is executed, steps that must be executed over the system and postconditions which must be fulfilled after the mentioned steps are carried out.

A validation test is considered to be successful if the postconditions hold when the test is over. In other cases, the validation test fails and the requirements it covers cannot be validated. In this case, some corrective actions are required so that the system can be validated and accepted by the user.

Finally, a traceability matrix is included in table 3.61, which shows the correspondence between user requirements and validation tests (this information is also shown in the tests tables). As it can be observed in the figure, the traceability matrix is complete, thus all the requirements are covered by at least one validation test and the validation plan is correct.

The validation tests will be executed after the system implementation is completed and the results will be analyzed in chapter 6.

ID	Test 01.
Name	Database connection.
Description	The system shows a screen for the user to establish a database connection (with a username and a password).
Pre-Conditions	• The user has started the application.
Steps	1. The system shows the database authentication screen. 2. The user inputs the database connection information and clicks on *Connect*. 3. The system validates the information and, if it is correct, it establishes the connection.
Post-Conditions	• A database connection is established. • The interface is enabled for the user to use the management and simulation tools.
Covered Reqs.	FR-01, NFR-SA-01.

Table 3.49: Validation test 01 (Database connection)

ID	Test 02.
Name	Situations management.
Description	The system shows a screen to manage (add or remove) situations for some fixed situational aspects.
Pre-Conditions	• The user has authenticated in the application. • The user selects the screen for situations management.
Steps	1. The system shows a list with the different situational aspects. 2. The user selects a situational aspect for the list. 3. The system shows a list with the situations for the selected situational aspect. 4. The user may perform two different management operations: (a) add a new situation for the selected situational aspect, given its name and description or (b) delete an existing situation for the selected situational aspect. 5. The system performs the user operation.
Post-Conditions	• Either the new situation is added to the list of situations, or the removed situation no longer exists in the system.
Covered Reqs.	FR-02, FR-04, FR-MA-24, FR-MA-25, FR-MA-26.

Table 3.50: Validation test 02 (Situations management)

ID	Test 03.
Name	Features management.
Description	The system shows a screen to manage (add or remove) features.
Pre-Conditions	• The user has authenticated in the application. • The user selects the screen for features management.
Steps	1. The system shows a list with all the stored features. 2. The user may perform three different operations: (a) add a new feature, (b) delete an existing feature or (c) edit the cost factors for an existing feature. 3. The system performs the user operation.
Post-Conditions	• Either the new feature is added to the list of features, the removed feature no longer exists in the system or the cost factors for the selected feature are updated.
Covered Reqs.	FR-02, FR-04, FR-MA-10, FR-MA-11, FR-MA-28.

Table 3.51: Validation test 03 (Features management)

ID	Test 04.
Name	Network selection.
Description	The system shows a screen to select an existing network, or create a new one.
Pre-Conditions	• The user has authenticated in the application. • The user selects the screen for network management or network simulation.
Steps	1. The system shows a list with all the existing networks. 2. The user may perform three different actions: (a) create a new network, identified by a name and a description, (b) remove an existing network or (c) select an existing network. 3. The system performs the user operation.
Post-Conditions	• Either the new network is created, the removed network no longer exists in the system or the system loads the selected network.
Covered Reqs.	FR-02, FR-MA-01, FR-MA-02, FR-MA-03, FR-SI-01.

Table 3.52: Validation test 04 (Network selection)

ID	Test 05.
Name	Plane selection.
Description	The system shows a screen to select a network plane.
Pre-Conditions	• The user has authenticated in the application. • The user has selected a network. • The user selects the screen for network management or network simulation.
Steps	1. The system shows a list of the existing planes for the selected network. 2. The user may perform three different actions: (a) create a new plane, identified by a name, (b) remove an existing plane or (c) load an existing plane. 3. The system performs the user operation.
Post-Conditions	• Either the new plane is created, the removed plane no longer exists in the network or the system loads the selected plane.
Covered Reqs.	FR-02, FR-MA-13, FR-MA-14, FR-MA-15, FR-SI-02.

Table 3.53: Validation test 05 (Plane selection)

ID	Test 06.
Name	Plane visualization.
Description	The system shows a network plane and its network items.
Pre-Conditions	• The user has authenticated in the application. • The user has selected a network and a network plane.
Steps	1. The system loads the plane and shows its background image and network items (nodes and links). 2. The user may perform the next operations: (a) load a new background image for the plane or (b) move and zoom through the plane. 3. The system performs the user operation.
Post-Conditions	• The plane is loaded and shown in the interface. • A new background image is loaded for the plane if the user selected that option.
Covered Reqs.	FR-02, FR-MA-14, FR-MA-23.

Table 3.54: Validation test 06 (Plane visualization)

ID	Test 07.
Name	Network items management.
Description	The system shows a screen to manage network items.
Pre-Conditions	• The user has authenticated in the application. • The user has selected a network and a network plane. • The user selects the screen for network management.
Steps	1. The user may perform the next operations: (a) create a new node or link, (b) remove an existing node or link or (c) edit the properties (description, coordinates, basic cost, activeness, features and cost factors) of an existing node or link. 2. The system performs the user operation.
Post-Conditions	• Either the new item is created, the removed item no longer exists in the network or the selected item properties are updated, according to the user actions.
Covered Reqs.	FR-02, FR-04, FR-MA-04, FR-MA-05, FR-MA-06, FR-MA-07, FR-MA-08, FR-MA-09, FR-MA-12, FR-MA-16, FR-MA-17, FR-MA-18, FR-MA-19, FR-MA-20, FR-MA-21, FR-MA-22, FR-MA-27.

Table 3.55: Validation test 07 (Network items management)

ID	Test 08.
Name	Agent simulation.
Description	The system simulates the motion of an agent over the network.
Pre-Conditions	• The user has authenticated in the application. • The user has selected a network and a network plane. • The user selects the screen for network simulation.
Steps	1. The user may perform the next operations: (a) place the agent over the network, (b) move the agent across the network using the mouse or keyboard or (c) use a physical layer simulation (GPS, RFID) to move the agent to a new position over the network. 2. The system places the agent in the new position.
Post-Conditions	• The agent is placed in a new position of the network. • The closest node is selected as the source node for the simulation services which the user may request.
Covered Reqs.	FR-03, FR-SI-03, FR-SI-04, FR-SI-05, FR-SI-10.

Table 3.56: Validation test 08 (Agent simulation)

ID	Test 09.
Name	Simulation services.
Description	The system shows a screen to simulate a situation model service.
Pre-Conditions	• The user has authenticated in the application. • The user has selected a network and a network plane. • The user selects the screen for network simulation. • The user has placed the agent over the network.
Steps	1. The user may perform the next operations: (a) select the description service, (b) select the routing service or (c) select the navigation service. 2. The system executes the requested service.
Post-Conditions	• Either the system describes the user location (description service) or starts the routing or navigation service.
Covered Reqs.	FR-03, FR-SI-06, FR-SI-07, FR-SI-08, FR-SI-09.

Table 3.57: Validation test 09 (Simulation services)

ID	Test 10.
Name	Routing and navigation services.
Description	The system shows a screen to simulate a situation model service.
Pre-Conditions	• The user has authenticated in the application. • The user has selected a network and a network plane. • The user selects the screen for network simulation. • The user has placed the agent over the network. • The user has requested the routing or navigation services.
Steps	1. The user may perform the next operations: (a) pick the target location (destination) for the routing or (b) choose applying situations for the simulation. 2. The system carries out the requested operation.
Post-Conditions	• Either the system fixes the selected goal or updates the applying situations. • The system shows the most adequate route from the source to the goal. • The system produces the output for the service (the route for the routing service and the next step for the navigation).
Covered Reqs.	FR-03, FR-SI-06, FR-SI-08, FR-SI-09, FR-SI-10, FR-SI-11, FR-SI-12.

Table 3.58: Validation test 10 (Routing and navigation services)

ID	Test 11.
Name	Big networks.
Description	The system is able to work with a big network.
Pre-Conditions	• The user has authenticated in the application. • The user has selected a network and a network plane.
Steps	1. The user may perform the next operations: (a) manage a network with more than 500 network items or (b) perform a simulation over a network with more than 500 network items. 2. The system carries out the requested operation.
Post-Conditions	• The management or simulation services are executed over the big network.
Covered Reqs.	NFR-SC-01.

Table 3.59: Validation test 11 (Big networks)

ID	Test 12.
Name	Interoperability
Description	The system is able to run in different platforms.
Pre-Conditions	*None*
Steps	1. The user starts the application either in Windows XP or higher, or in Linux systems.
Post-Conditions	• The application starts and is operable.
Covered Reqs.	NFR-IO-01.

Table 3.60: Validation test 12 (Interoperability)

Table 3.61 shows the traceability matrix, which graphically plots how the requirements are covered by each validation test.

	01	02	03	04	05	06	07	08	09	10
FR-MA-20							x			
FR-MA-19							x			
FR-MA-18							x			
FR-MA-17							x			
FR-MA-16							x			
FR-MA-15					x					
FR-MA-14					x	x				
FR-MA-13					x					
FR-MA-12							x			
FR-MA-11			x							
FR-MA-10			x							
FR-MA-09							x			
FR-MA-08							x			
FR-MA-07							x			
FR-MA-06							x			
FR-MA-05							x			
FR-MA-04							x			
FR-MA-03				x						
FR-MA-02				x						
FR-MA-01				x						
FR-04		x	x				x			
FR-03								x	x	x
FR-02		x	x	x	x	x	x			
FR-01	x									

	01	02	03	04	05	06	07	08	09	10	11	12
NFR-IO-01												x
NFR-SA-01	x											
NFR-SC-01											x	
FR-SI-12										x		
FR-SI-11										x		
FR-SI-10								x		x		
FR-SI-09									x	x		
FR-SI-08									x	x		
FR-SI-07									x			
FR-SI-06									x	x		
FR-SI-05								x				
FR-SI-04								x				
FR-SI-03								x				
FR-SI-02					x							
FR-SI-01				x								
FR-MA-28			x									
FR-MA-27							x					
FR-MA-26		x										
FR-MA-25		x										
FR-MA-24		x										
FR-MA-23						x						
FR-MA-22							x					
FR-MA-21							x					

Table 3.61: Traceability matrix

3.6 Development Methodology

The software development process for the project will obey the spiral lifecycle model [Boehm, 1986]. This development model combines the simplicity and formality of the waterfall model with the flexibility of prototyping approaches, and turns out to be a good approach for research projects where a complete set of requirements is hardly known from the very beginning and thus may require several prototypes before achieving a satisfactory result, like the tool which is object of this project. Figure 3.2 shows the concept of the spiral model graphically.

The spiral model provides four main phases which are repeated for each iteration. These phases are the next ones:

1. **Planning**: during this phase, the objectives for the current iteration are determined, and a requirements analysis is performed. Each iteration will refine the requirements eventually leading to the complete requirements set for the software application.
2. **Analysis and Design**: this phase performs a risk analysis and a feasibility study. During this stage, a feasible prototype will be generated.
3. **Development**: during this phase, the prototype is designed, implemented and tested.
4. **Evaluation**: this phase evaluates the already generated prototype. The main purpose of this evaluation is to detect the defects of the prototype in order to be used when planning planning the objectives for the next iteration.

The software development will be divided in three cycles, each of them providing as a result a new prototype of the application. Prior to the development, the tasks which will be performed during each phase will be estimated. The purpose of the first phase is to develop a prototype which provides basic network management features. During the second phase, a new prototype will be generated which will provide simulation features for the situation model. Finally, in the third cycle, the last prototype will support the management of situations and the simulation over them. This last cycle is also intended to perform a final refinement of the application, by detecting and implementing features that were not specified in the previous phases.

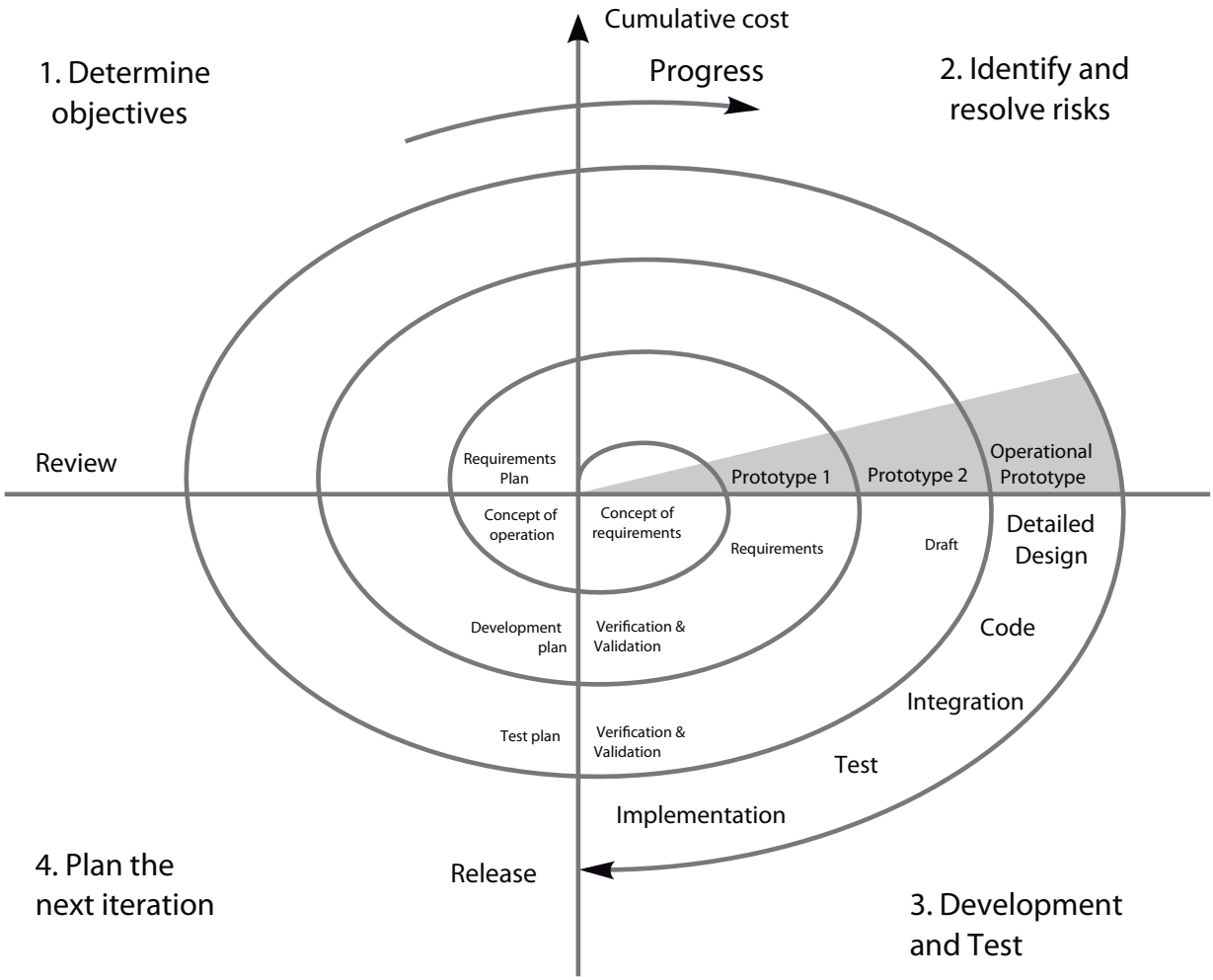


Figure 3.2: Spiral lifecycle development model

3.7 Project Planning

This section details the projected planning, both in terms of schedule (temporal planning) and budget (costs planning).

3.7.1 Time Estimation

Figure 3.3 shows a Gantt chart with the estimated planning for each task. For this planning, it has been taken into account that there is only one developer which is working in the project 20 hours per week (i.e. 4 hours per day, 5 days per week).

The estimated time for the whole development process is of two months and two weeks

(starting on March 12th and ending in May 25th). Additionally, two extra months are considered for non-development tasks, such as research on the state-of-the-art and requirements elicitation and specification; thus considering a total project time of four months and two weeks.

Id.	Task	Start	End	Duration
1	**Iteration 1**	**12/03/2012**	**20/04/2012**	**30d**
2	Planning	12/03/2012	12/03/2012	1d
3	Analysis	13/03/2012	14/03/2012	2d
4	Design	15/03/2012	16/03/2012	2d
5	Development	19/03/2012	16/04/2012	21d
6	Testing	17/04/2012	19/04/2012	3d
7	Documentation	20/04/2012	20/04/2012	1d
8	**Iteration 2**	**23/04/2012**	**09/05/2012**	**13d**
9	Planning	23/04/2012	23/04/2012	1d
10	Analysis	24/04/2012	24/04/2012	1d
11	Design	25/04/2012	26/04/2012	2d
12	Development	27/04/2012	04/05/2012	6d
13	Testing	07/05/2012	08/05/2012	2d
14	Documentation	09/05/2012	09/05/2012	1d
15	**Iteration 3**	**10/05/2012**	**25/05/2012**	**12d**
16	Planning	10/05/2012	10/05/2012	1d
17	Analysis	11/05/2012	11/05/2012	1d
18	Design	14/05/2012	15/05/2012	2d
19	Development	16/05/2012	22/05/2012	5d
20	Testing	23/05/2012	24/05/2012	2d
21	Documentation	25/05/2012	25/05/2012	1d

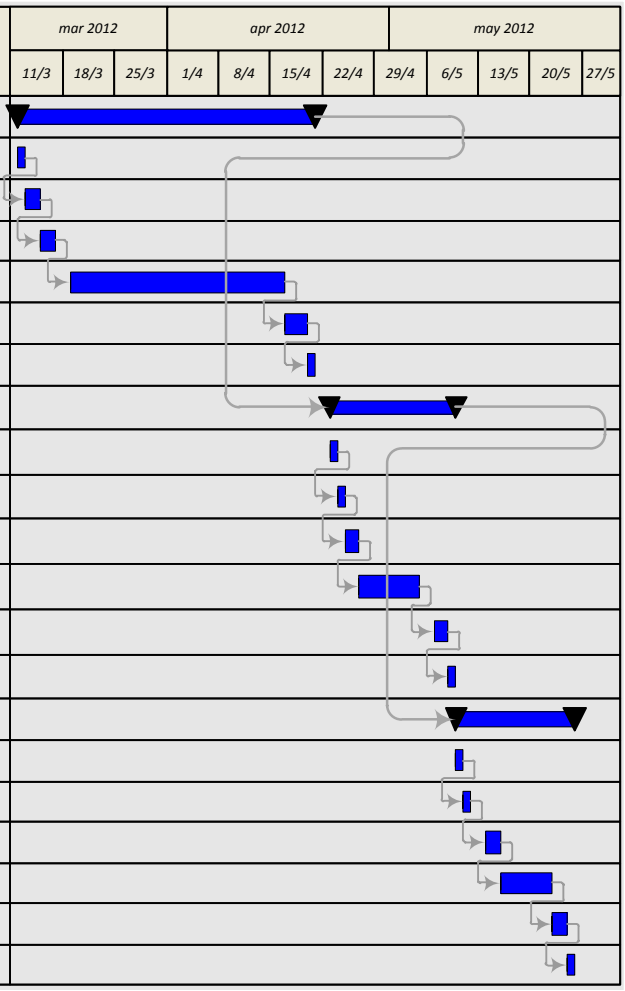


Figure 3.3: Temporal planning for the development (Gantt chart)

3.7.2 Costs Projection

The costs projection for this project are detailed in this section.

Table 3.62 shows the detailed estimated costs for physical resources. Regarding the technological resources, computer servers are considered to have an average lifetime of 5 years (depreciation of 20% per year), while for personal computers the average lifetime are 2 years (depreciation of 50% per year); and none of them are considered to have any residual value when their lifetime expires.

Qty.	**Concept**	**Unitary cost**	**Total**
1	Computer Server (Sun Fire X4150 Server)	2450 €	165 €
1	Personal Computer (Intel Quad Core, 4GB RAM)	650 €	110 €
1	Fungible goods	60 €	60 €
			327 €

Table 3.62: Costs projection for physical resources

Table 3.63 shows the detailed estimated costs for human resources, based on the time planning estimated on the previous section.

Profile	**Cost per hour**	**Hours**	**Total**
Project Manager	35 €	60 hours	2100 €
Technical Consultant	25 €	40 hours	1000 €
Requirements Engineer	20 €	40 hours	800 €
Analyst Developer	25 €	180 hours	4500 €
Tester	20 €	20 hours	400 €
		340 hours	**8800 €**

Table 3.63: Costs projection for human resources

Finally, table 3.64 summarizes the total costs projection for the entire project.

Concept	**Total**
Physical resources	327 €
Human resources	8800 €
	9127 €

Table 3.64: Total costs projection

Chapter 4

Analysis and Design

The purpose of this chapter is to provide a high-level abstraction of the software design given the requirements specification from chapter 3, in order to provide a solution which covers these requirements. Sections 4.1 and 4.2 describe, respectively, the physical and the functional architectures of the system. The latest one includes the system data and the system logic, which is divided into subsystems, and the integration of these subsystems.

After that, section 4.3 describes the design for the databases, including the different entities and the relationships among them.

Finally, section 4.4 shows a high-level design for the user interfaces, describing the components for each of the screens.

4.1 Physical Architecture

The physical architecture for the software is based on a client-server model. The server (or servers) will provide two differect roles: the storage of the situation model and the storage of the edition information for the cognitive architecture (Cognos). The clients will connect to the servers both for the use and the management of the situation model. It must be pointed out that this architecture requires the servers to be accessible by all the clients.

Actually, the server which stores the situation model is replicating a subset of the edition data in the Cognos server. This fact provides a natural redundancy for the model data. However, external mechanisms must guarantee the consistency of this replica, to ensure that the modifications performed using the management tool are also carried out in the situation model.

Some advantages of this physical architecture is that it supports concurrent edition and operation over the situation model (thus, the clients can take advantage of this parallelism), high scalability, confidentiality and reliability; as long as the database management system

guarantees these principles.

Figure 4.1 shows the physical architecture for the system graphically.

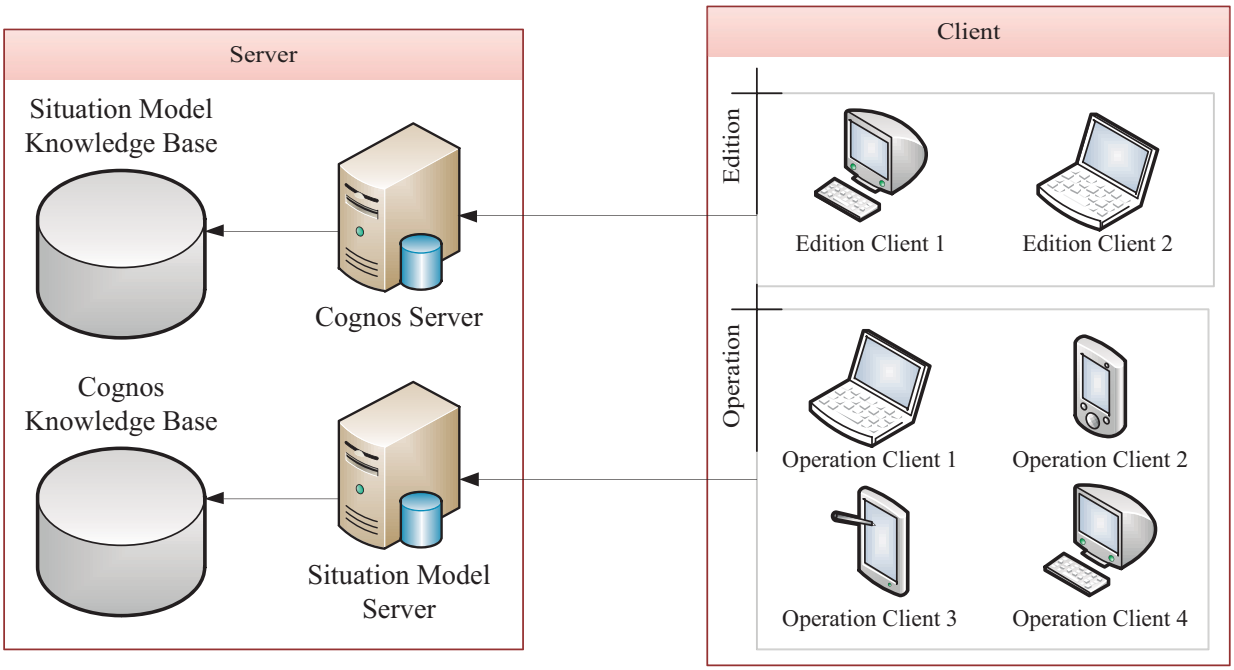


Figure 4.1: Physical client-server architecture

4.2 Functional Architecture

Once the physical architecture is defined, the next step proposes a design for the functional architecture of the system.

The system will follow a three-tier architecture [Eckerson, 1995]. This model develops the user interface (presentation tier), the process business logic (logic tier) and the data (data tier) as independent modules. However, although functionality is thus divided into three tiers, this functional vision does not affect the design of storages or the interaction.

This modularity provides several advantages, as any of these modules can be updated and maintained independently from each other; reducing the coupling between the different components. For example, this model would allow the use of different user interfaces, designed for different computer devices, using the same application logic and the same data.

Additionally, decoupling the data from the rest of the application also improves the system performance and scalability.

Figure 4.2 provides an overview of the application design, showing the subsystems for each of the three tiers.

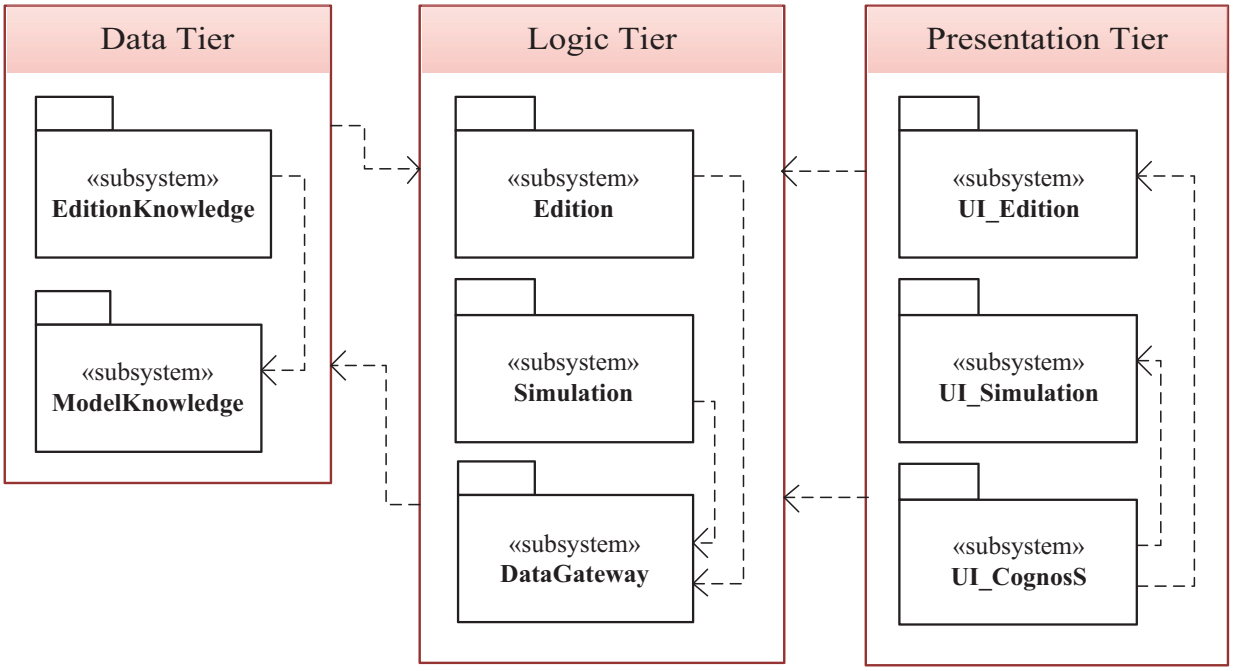


Figure 4.2: 3-tier architecture design

Throughout this section, it is provided a more detailed description for each of the tiers and its components.

4.2.1 Data Tier

The data tier contains the resources which serve for persistent storage, i.e. the mechanisms for data storage themselves (for example, a database management system), as well as the structures which may be required for handling the data.

For this project, a relational database management system will be used to satisfy the application persistent storage necessities. The choice of a relational database over other possible alternative was motivated by the predominance of these systems and their facility for the design and implementation, as well as for the ease when executing queries and updates over them.

The data tier will be accesible by a data controller in the logic tier, which will be described in the next section. As it is shown in figure 4.3, the data tier contains two subsystems: the model knowledge subsystem and the edition knowledge subsystem.

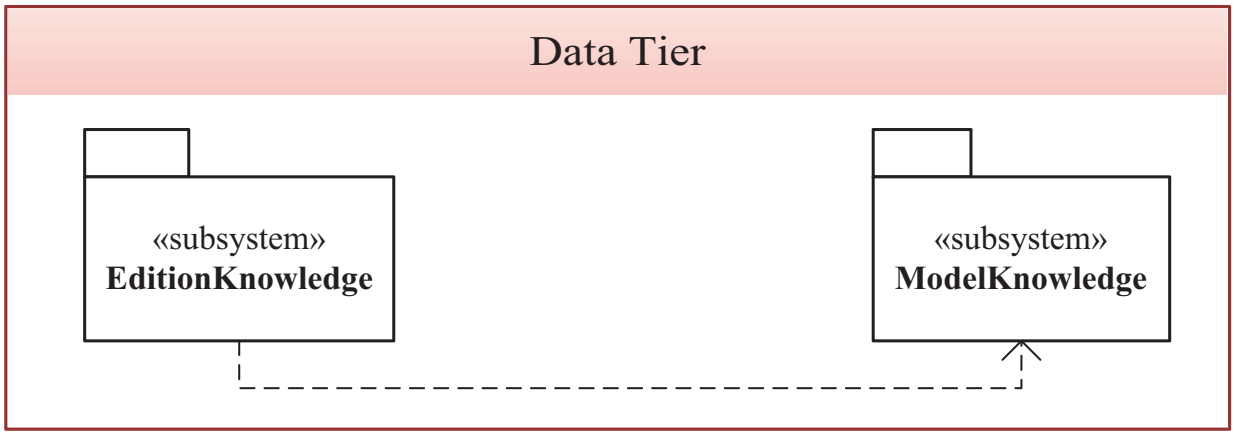


Figure 4.3: Subsystems for the data tier

4.2.1.1 `ModelKnowledge` Subsystem

The `ModelKnowledge` subsystem contains all the data which is required by the situation model. This data contains all the network-related information: nodes and links, costs, features and situations. This subsystem is intended to be able to work separately from the `EditionKnowledge` subsystem.

4.2.1.2 `EditionKnowledge` Subsystem

The `EditionKnowledge` subsystem contains additional information which, complemented with the model knowledge, supports the model edition functionality. Most of the data entities within this subsystem have the purpose of graphically representing the model data, such as nodes and links. For this reason, this subsystem considers a *plane* as a bidimensional entity to represent a subset of a network in the application.

4.2.2 Logic Tier

The logic tier gathers the business logic that provides the system functionality. It also acts as an intermediate layer between the data model and the user interface.

The data tier contains three different subsystems: the edition subsystem, the simulation subsystem and the data gateway. Figure 4.4 shows this structure graphically.

4.2.2.1 `Edition` Subsystem

The aim of the edition subsystem is to support the functionality for situations and features management and network edition. The software requirements for this functionality

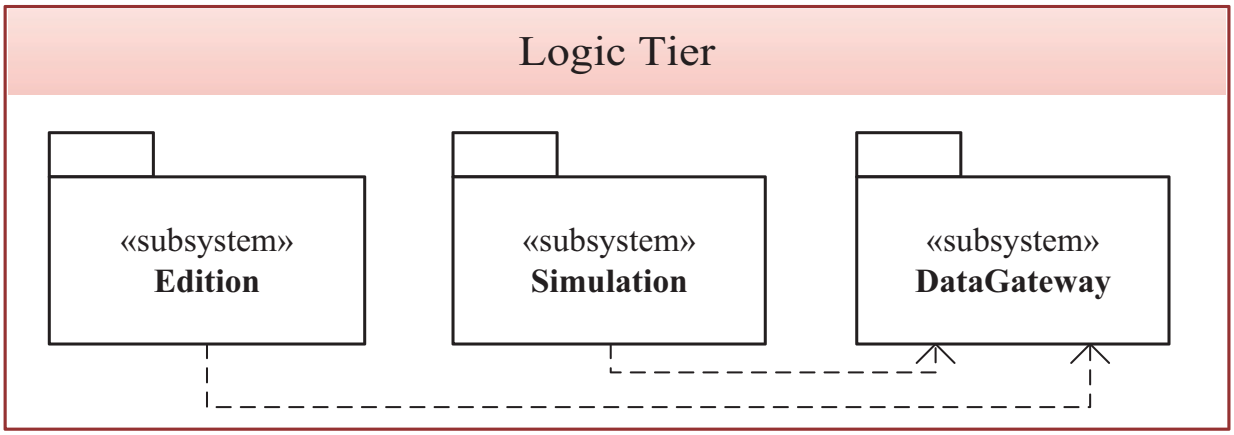


Figure 4.4: Subsystems for the logic tier

are detailed in section 3.2.1.1. In brief, this subsystem must provide logic for the next operations:

- Management of situations, this is, creating and removing situations which could apply for the simulation.
- Management of item features and cost factors, by creating and removing features, as well as assigning cost factors to the network items. These cost factors could be applied directly or by means of features.
- Network edition, which includes the selection of a network, the selection and management for planes and the management of network items (nodes and links) and its properties.

4.2.2.2 `Simulation` Subsystem

The simulation subsystem is intended to support the simulation functionality, whose software requirements are detailed in section 3.2.1.2. Thus, this subsystem must provide logic for the next operations:

- Selection of a network, as well as the network plane where the simulation will start.
- Selection of applying situations for the simulation, which will dynamically change the cost for each network item.
- Motion of the agent over the network, through direct motion or by means of a physical layer.
- Selection of a simulation service: description, routing or navigation, as well as additional requirements for the service execution (e.g. a target node for the routing or navigation service).

- Execution of the selected service and generation of the service output.

4.2.2.3 `DataGateway` Subsystem

The data gateway is an independent module which performs logic operations which deals with the data tier directly. For this case, the data gateway will provide routines to perform Create, Read, Update and Delete (CRUD) operations over a database.

The inclusion of a data gateway as a separate module provides several advantages, as it increases modularity and decreases the coupling. Actually, the modules in the simulation subsystem always perform the database communication by means of the data gateway, using the methods provided by this subsystem. For this reason, any change over the database design will require only the data gateway to be updated in order to comply with the new design, while other modules can still rely on the gateway subsystem to perform their data CRUD operations.

4.2.3 Presentation Tier

The presentation tier contains the modules in charge of handling the user interfaces which are required to interact with the system. In this case, the presentation tier includes three different subsystems, as shown in figure 4.5. This section describes each of them, providing an approach of the GUI controls that they must provide to support the required functionality.

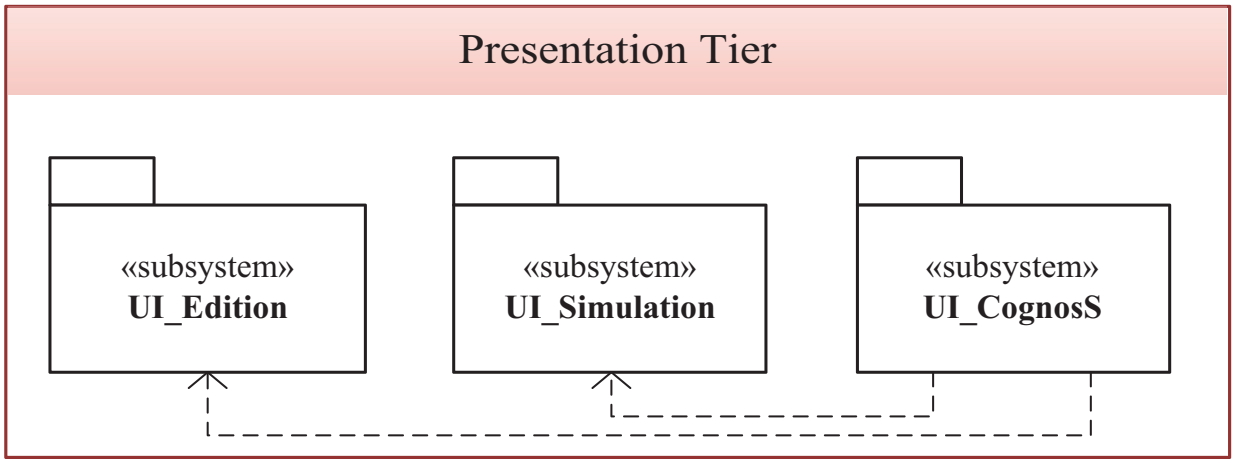


Figure 4.5: Subsystems for the presentation tier

4.2.3.1 `UI_CognosS` Subsystem

This subsystem contains the main frame of the application. It serves as an entry point for all the system functionality. Additionally, this subsystem also provides an interface which allows the user to authenticate and establish a connection to a database containing the knowledge model.

4.2.3.2 `UI_Edition` Subsystem

This subsystem provides the interfaces which are required for carrying out the management operations. These operations can be divided into the situations and features management and the networks management. Given that the network management may require a very complex interface, two different screens will be used for this subsystem: the first one for features and situations management and the second one for network edition.

4.2.3.3 `UI_Simulation` Subsystem

This subsystem provides an interface which serves for running a simulation over the situation model. The simulation will consist on an agent which will move across the network while a requested service (such as navigation) is executed.

4.3 Data Storage

Two different data stores are required for this project. The first store contains the knowledge base of the situation model. Meanwhile, the second store keeps all the data which is required to manage the model, i.e., it contains all the entities which are required to perform an edition operation over the situation model

4.3.1 Model Knowledge Base

Figure 4.6 shows an ER model which shows the conceptual design for the model knowledge store.

As figure 4.6 shows, networks and situations are the only entities which do not require any other entity to exist. A network is composed by network items, which could be either nodes or links between two nodes. It may be also noticed that the mechanism to assign costs to the network items is redundant, as these costs can be directly assigned to the network items, or assigned by means of items features.

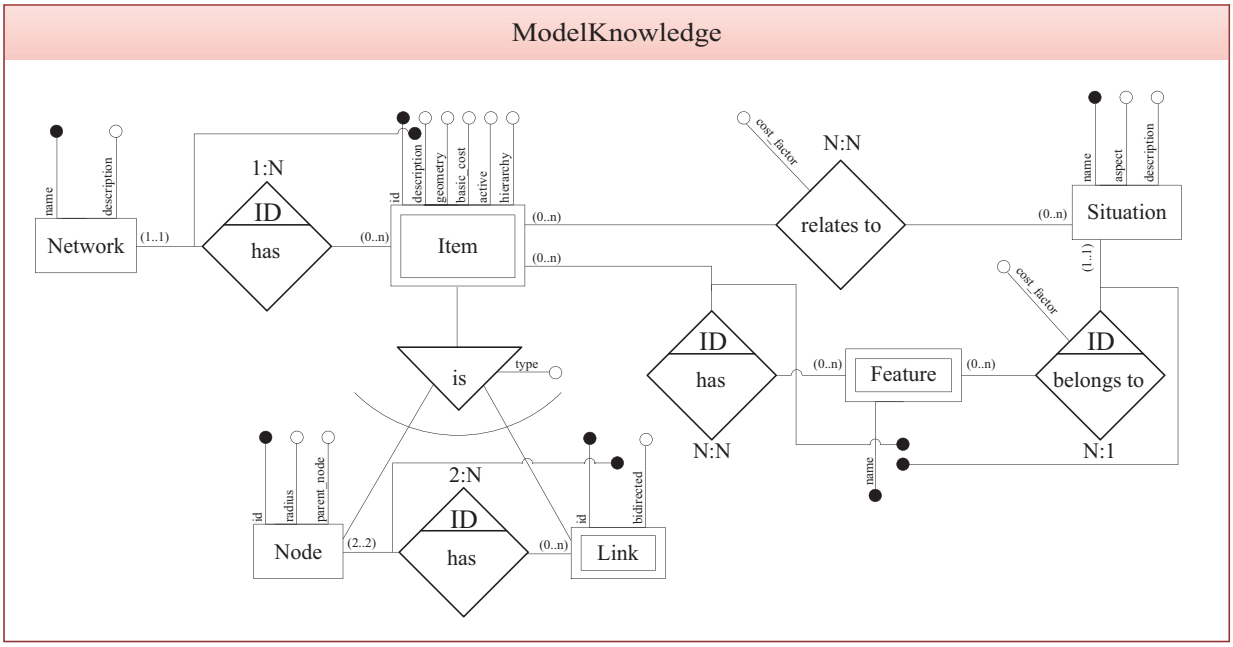


Figure 4.6: ER model for the model knowledge

4.3.2 Edition Knowledge Base

Figure 4.7 shows an ER model which serves as a conceptual representation of the data which is required by the system to operate. The diagram distinguishes between the model knowlegde and the edition knowledge entities. Note that the data store of the edition knowledge requires both of them: while the situation model knowledge can be stored and operated independently, the edition knowledge requires the existence of the model knowledge entities.

Then, the edition database is actually a superset of the model knowledge base. Keeping the same structure in the two data stores provides some advantages. In the first place, extracting the knowledge from the edition database is a trivial task, as no translation between different architectures is required. Secondly, the edition database would support the simulation over it.

Regarding the edition knowledge, each network entity in the model has an associated entity in the edition database, which serves to draw a graphical representation of that network item. However, the entity *Plane*, which does not exist in the model knowledge, is required in order to draw a multidimensional network in a two-dimensional grid, for the edition purposes. *VisualNode* and *VisualLink* entities are used for drawing the network.

The reader may have noticed that a loop exists between the entities *VisualNetwork* - *Network* - *Item* - *Node* - *VisualNode* - *Plane* - *VisualNetwork*. Although this could lead to eventual inconsistencies, it can be easily proved that any of these relations can be actually removed. As the model knowledge may exist independently, the relationships within that model must be kept. Regarding the other relations:

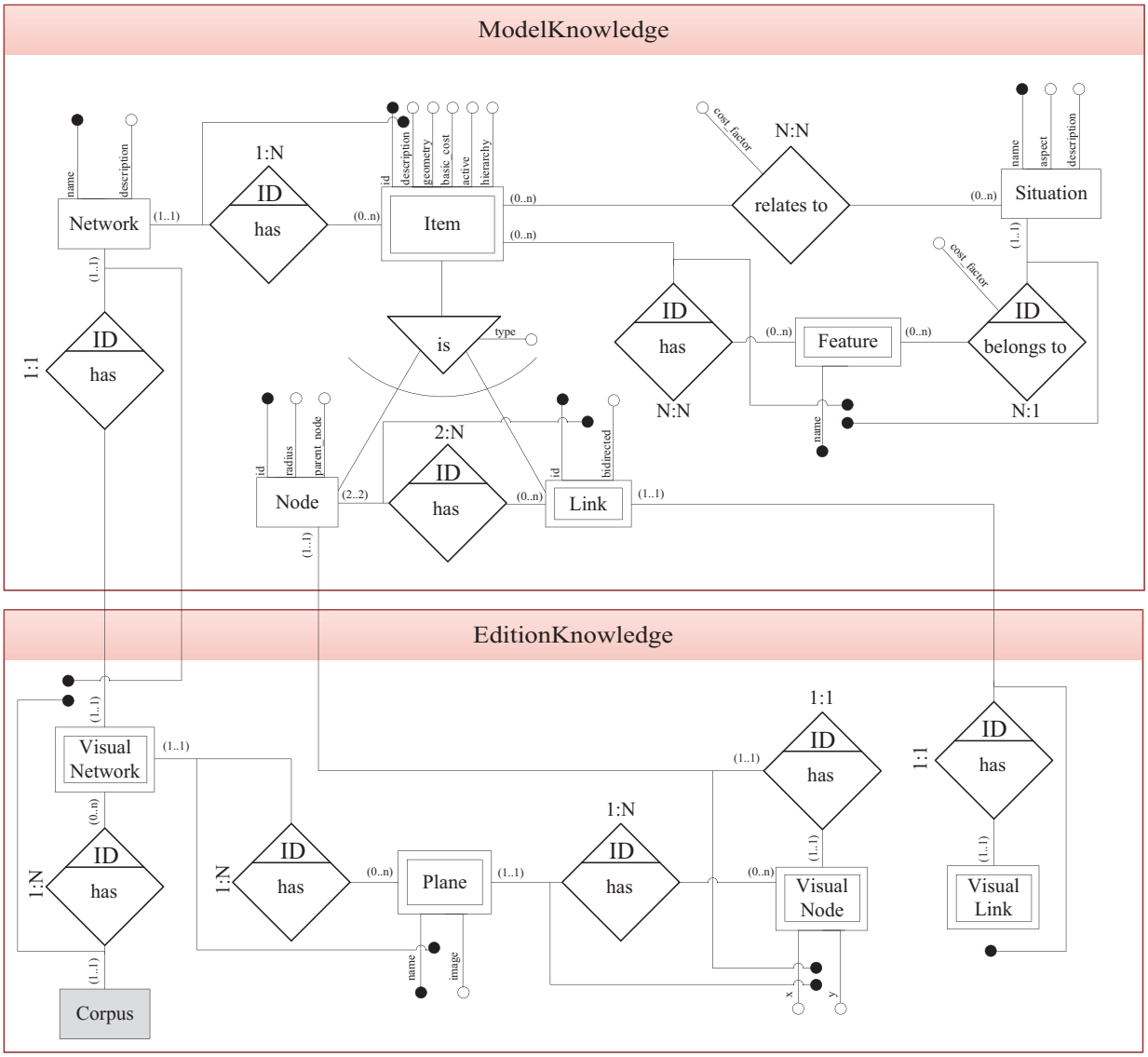


Figure 4.7: ER model for the edition knowledge

- The *VisualNode* - *Node* relationship must exist, as there are not any other way of relating a node from the model knowledge with a visual node from the edition knowledge.

- The *Plane* - *VisualNode* relationship cannot be removed, as there are not alternative ways of specifying which nodes belong to a given plane.

- The *VisualNetwork* - *Plane* relationship could be removed if it could known which planes belong to a network given the nodes belonging to that plane. Although it can be actually knownn, it is required that the plane contains at least one node, which will not always happen. Thus, the relationship must exist.

- A similar reasoning can be applied to the *Network* - *VisualNetwork* relationship.

Given that the model could theoretically lead to some inconsistencies, the edition tool must provide mechanisms for controlling the redundance, thus preventing these inconsistencies and avoiding incoherence. Moreover, some additional mechanisms may be implemented in the database with the same purpose, such as triggers or check constraints.

The *Corpus* entity, which is emphasized in the edition knowledge data, serves as a connection point with the existing Cognos architecture, and serves for integration purposes.

4.4 Interaction Design

This section goes deeper by providing some mockups of the different screens of the GUI, which describe a high-level design of the interfaces that will be shown to the user, and which will be used to operate the application.

Figure 4.8 shows a mockup of the main screen, corresponding to the `UI_CognosS` subsystem. This screen allows to start a connection to a knowledge base, and serves as an entry point for the other application modules through the tab panel selector at the top.

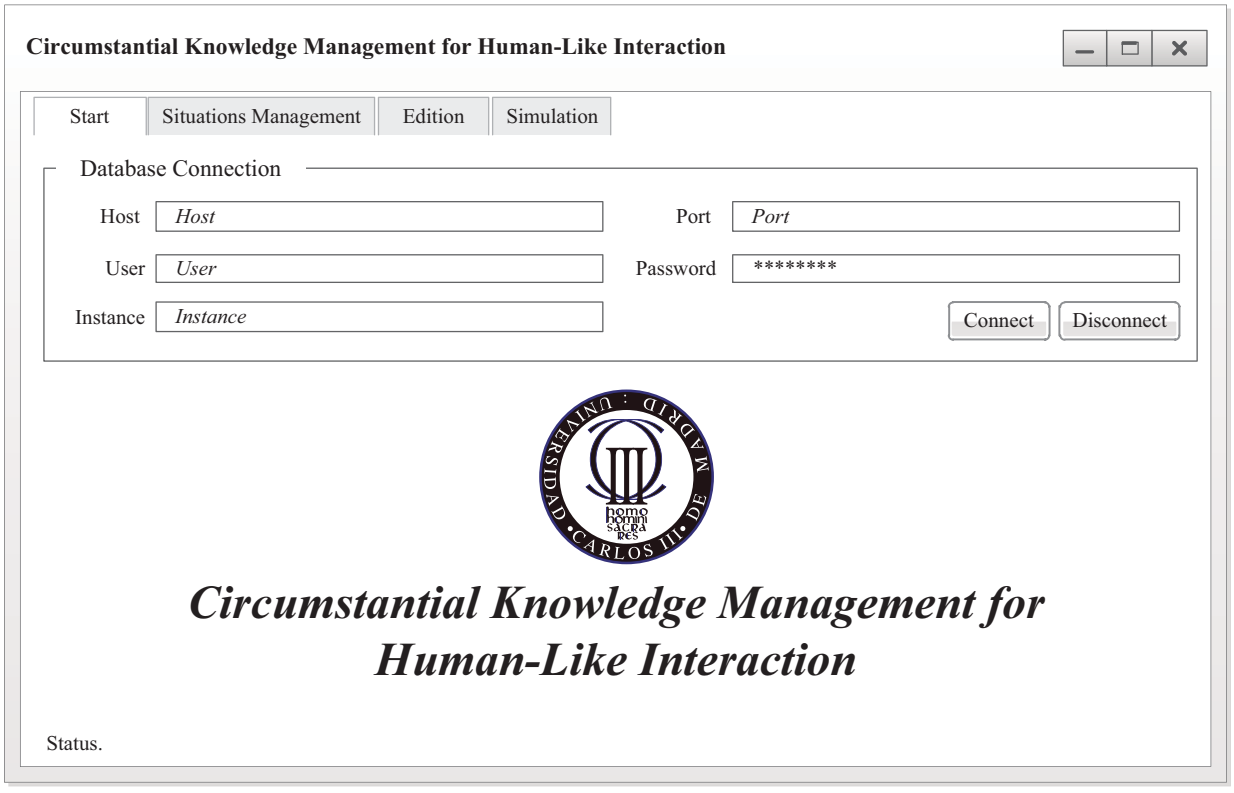


Figure 4.8: Mockup for the application start screen

In brief, this subsystem will contain one screen, which must show the next components:

- A way to go into the other interface components. A good approach for this mechanism could be a tab selector, where the different application functionalities are placed in different tabs. This provides an intuitive way for navigating between the different subsystems. Additionally, some tabs may be disabled if they are not accessible at any given moment in time, so the user is able to recognize anytime the operations that can be performed.
- A database connection form, which would contain all the required fields to establish the connection to the knowledge base, and also some mechanism to close the connection.

As these components require little space, it has been found appropriate to place in this screen the *about us* form which shows the logo and the name of the application.

Finally, a status bar is shown at the bottom, providing some feedback to the user on whether the connection to the database was correctly established or an error occurred.

Figure 4.9 shows the mockup for the first screen of the *UI_Edition* subsystem, which allows situations and features management. This screen would contain different panels, to add and remove situations and features, as well as to assign cost factors to the features.

As this screen will provide means to manage situations and features, it requires the next components:

- A field to list the existing situations in the system. The contents of this list may be filtered by a selected *situational aspect*. When a situation from the list is selected, its description will be shown, so that the user can check that the desired network is the selected one. Moreover, a button will allow to remove a selected situation.
- A *new situation* form, which will contain the required fields to add a new situation to the system.
- A field to list the existing features in the system. A button will allow to remode a selected feature.
- A *new feature* form, which will allow the required fields to add a new feature to the system.
- A *feature edition* form, which will contain a field to assign cost factors to a selected feature.

On the other hand, figure 4.10 shows a mockup for the second screen of the *UI_Edition* subsystem, which allows networks management. This screen contains panels for networks

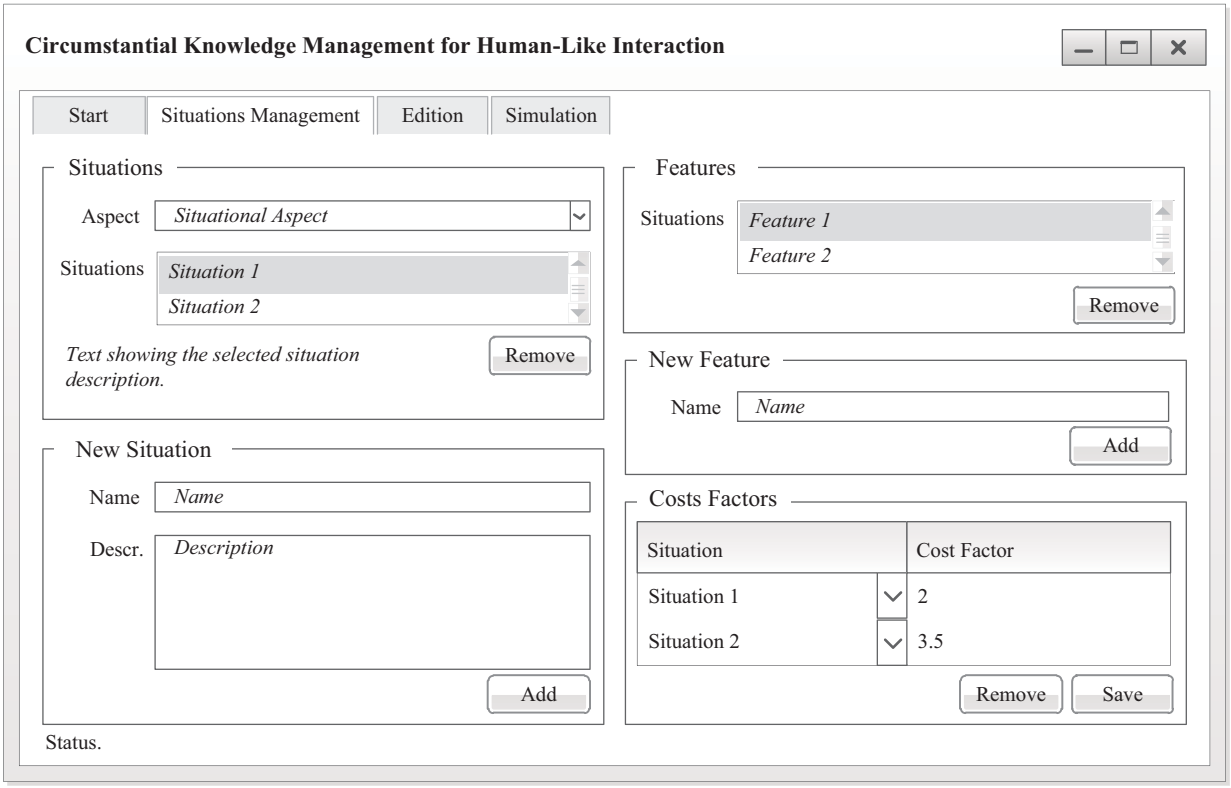


Figure 4.9: Mockup for the situations and features management screen

and planes creation, deletion and selection. Once a network and a plane are selected, the plane image is shown and the network edition can be performed over it, adding or removing nodes and links, as well as editing its properties.

The second screen would contain all the controls to allow the networks management:

- A chooser element to select an existing network in order to load it, as well as a text to show the description for the chosen network. Additionally, a button must allow the removal of the selected network.
- A *new network* form with all the required fields to create a new network.
- A chooser element to select a plane for the selected network, and a button to remove the selected plane.
- A *new plane* form with all the required fields to create a new network plane.
- A panel showing the image for the selected network plane. A button to load a new image into the plane. A slider to zoom over the image.
- A set of buttons to add and remove network items, as well as to select network items to be edited.

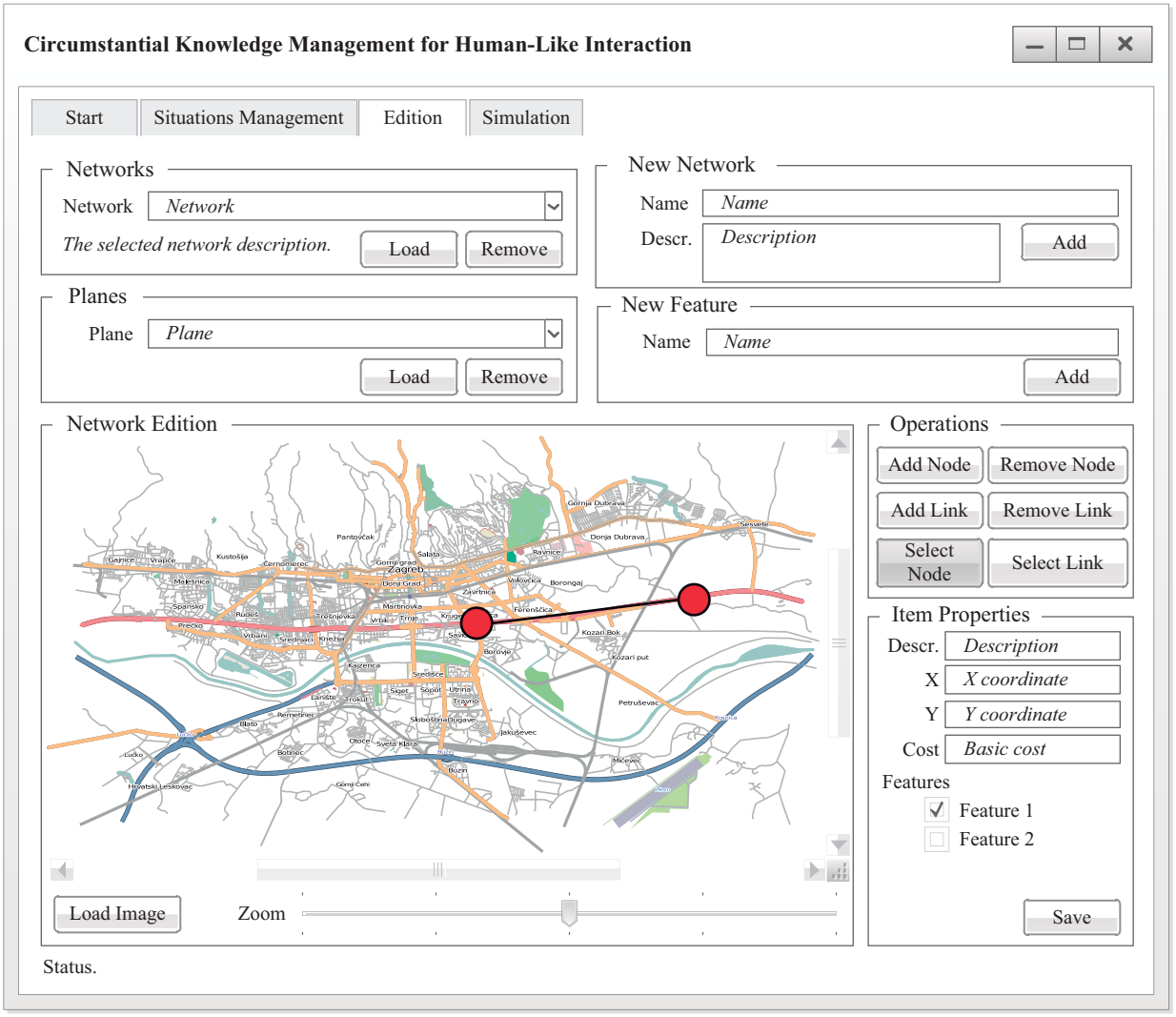


Figure 4.10: Mockup for the networks management screen

- A *network item edition* form with all the required fields to edit a selected network item.

Finally, figure 4.11 shows a mockup for the simulator screen. This screen contains all the elements described above, which are required to perform a simulation over the situation model.

The simulation screen would require the next components:

- A chooser element to select an existing network in order to load it, as well as a text to show the description for the chosen network..
- A chooser element to select a plane for the selected network.

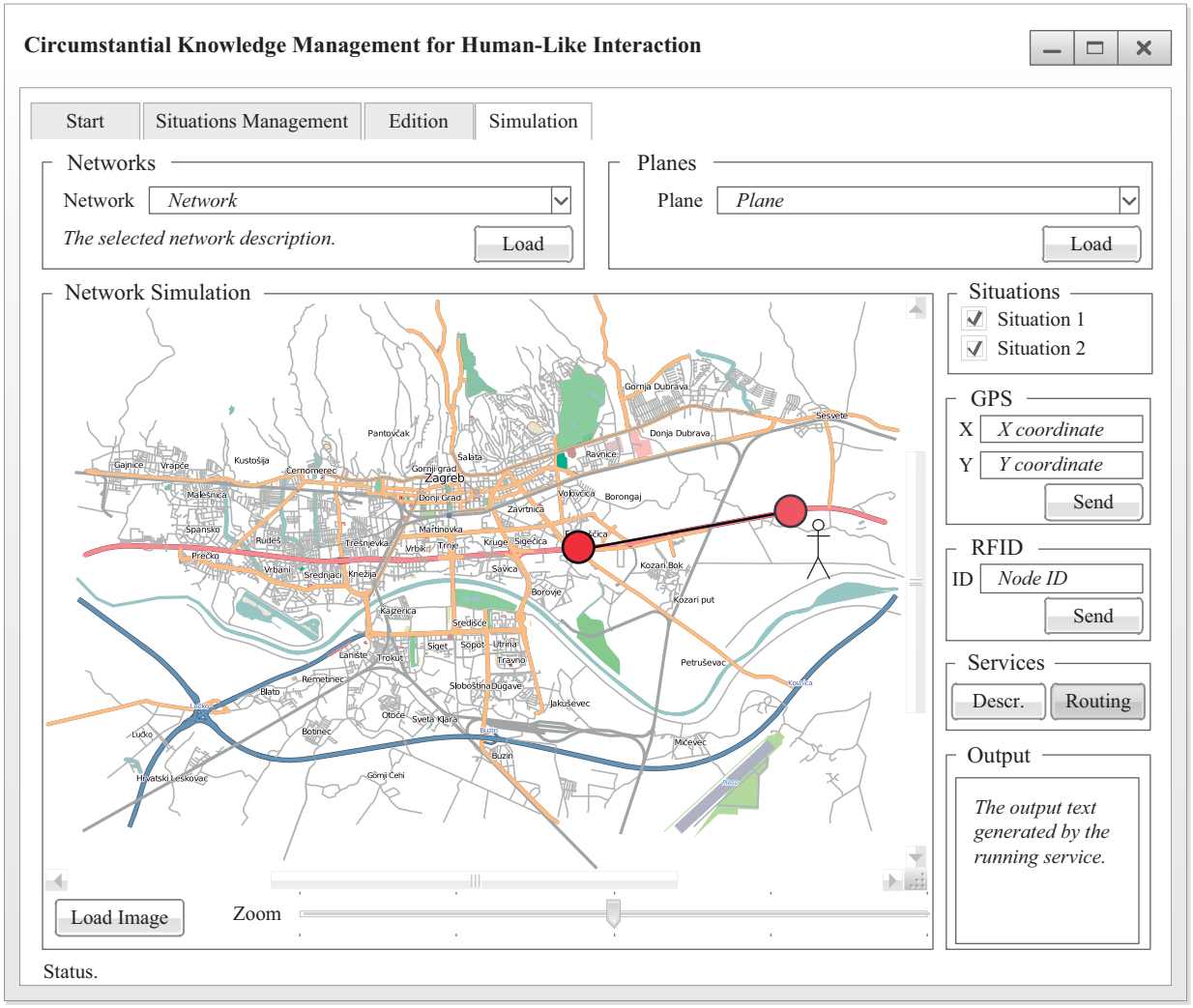


Figure 4.11: Mockup for the simulator screen

- A panel to display selected network plane. A slider to zoom over the image.
- An image representing the agent, which can be placed over the plane image.
- Mouse and keyboard events required to place and move the agent over the network.
- A panel for each physical layer emulator. For the mockup construction, GPS and RFID as physical layer emulators will be used.
- A button for each service provided by the simulator. Some services may require some special controls.
- A list of situations, which can be selected to apply during the simulation.
- A text box to generate the services output.

Chapter 5

Implementation

The aim of this chapter is to delve into the design of the system, thus providing enough detail as required to serve as a guide for its implementation. This implementation must satisfy the physical and functional needs listed in the previous chapter.

Section 5.1 describes some considerations regarding the system implementation, which may be helpful to understand some implementation decisions of the later sections.

Moreover, sections 5.2 and 5.3 detail the implementation of the physical architecture for the system and provide a low-level design of the functional architecture, respectively. The contents of the latest one can be used directly for the system coding.

Finally, section 5.6 introduces the implementation of the data storages of the system.

5.1 Implementation Issues

This section describes some implementation considerations which may be helpful to understand some decisions carried out during system implementation.

The system implementation will follow an object-oriented approach. This turns out to be a good option as the system is structured into classes, each of them having well defined responsibilities. Moreover, it provides some features such as encapsulation, inheritance and polymorphism, which can be really useful for our purposes:

- The principle of encapsulation allows a class to hide its own attributes. Instead, public methods describes the functions which can be executed over an instance of the class (also known as an object). This provides modularity, and allow to use classes as black boxes, where nothing but its functionality is required to be known.
- The principle of inheritance allows to create a class hierarchy, where some classes are *childs* of other classes. Often, these *child* classes provides some specialization

over their *parents*, or some additional features. Moreover, abstract classes can be created, which cannot be instantiated directly. Thus, a hierarchy with several levels of abstraction can be built by means of inheritance.

- The principle of polymorphism allows to hide the actual type of the object. The variable type can be known for being an interface, or a *parent* class in a hierarchy. If the actual type of the instance complies with the known type, then its methods can be called.

Once the object oriented paradigm has been chosen as the preferred programming paradigm for this application, the next step consists on choosing the programming language for the system implementation. Although there are several alternatives of object-oriented languages, we have considered Java to be the most appropriate, as it was stated in the feasibility study (section 3.4). Besides the advantages listed in that section, many others can be found. First of all, it is highly extended, and the development team is already experienced in Java programming. Secondly, it executes over a virtual machine (Java Virtual Machine), and it provides multiplatform features. Thirdly, there are many libraries for Java which could already provide some functionality required by this system, and could be reused. Finally, there are many Integrated Development Environments (IDEs) which allows a What You See Is What You Get (WYSIWYG) creation and edition of user interfaces, which will be useful for the design of the application screens. On the other hand, Java is less efficient that other object oriented languages; however, this is not a big handicap as the big load of the application is intended to be carried out in the server side.

For the application development, Oracle NetBeans 7 will be used. This *IDE* for Java allows an easy-to-use, WYSIWYG editor for user interfaces. Moreover, it also supports collaboration tools, such as software versionining and revision control systems. For our case, Subversion will be used as a system for revision control and versioning, which also serves for storing a backup of each different version of the application source code.

Regarding the data storage, it has been decided to use a relational database management system. This decision was taken as the developers had previous experience on the design and development of relational databases. The chosen database management system is Oracle Database 11g, as it was already stated in the feasibility study, as licenses were provided by the LaBDA research group, and it can be easily integrated with Java to execute CRUD operations from the application.

5.2 Physical Architecture

Section 4.1 specified the physical architecture to be implemented for the system, which was a client-server model.

Actually, this architecture have been implemented using only one server, which was provided by the LaBDA group. This server contains two databases: the situation model

knowledge and the edition knowledge (Cognos). Moreover, one desktop computer is being used as the development computer and also as a client running the edition and simulation application. The technical specifications for these machines were specified in section 3.4.

5.3 Functional Architecture: Data Tier

This section will detail the different classes of the data tier. Additionally, it will include a relational graph which can be used for the implementation of the database model.

5.3.1 `ModelKnowledge` Subsystem

Figure 5.1 displays the class diagram for the classes within the `ModelKnowledge` subsystem. It is important to notice that from now on, the diagrams will not show the getters and setters methods, as all fields in a class are private and therefore they are supposed to have these accesor methods by default.

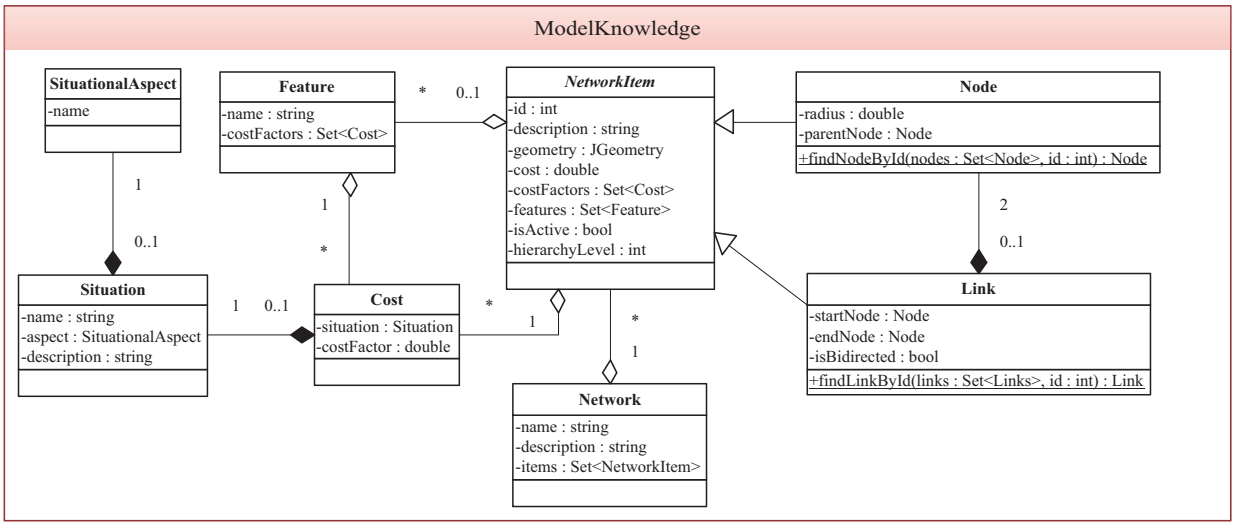


Figure 5.1: Classes for the `ModelKnowledge` subsystem

- The `SituationalAspect` class has only a name, which represents the situational aspect itself. As the different situational aspects were defined in the requirements and were a reduced set of values, we will implement this class as a Java enumerated type, which will contain the different possible values for the situational aspects.
- The `Situation` class is identified by a name and a situational aspect, and can also contain a description. It may be noticed that we have preferred to set the situational aspect as a field of the `Situation` class, rather than considering a set of situations for

each situational aspect. This decision makes it possible to use an enumerated type for the situational aspect.

- The `Network` class is identified by a name, and may also contain a description. Moreover, it contain a set of network items.
- The `NetworkItem` abstract class stores the common fields for both nodes and links. These items are identified by a positive integer, and may also contain a description. All the network items contains a geometry, which is a spatial type which may be useful for operating with these items within a spatial database management system. The documentation for this type can be found in [Oracle, 2012]. The network items also contain a basic cost, which can be complemented with additional cost factors. Moreover, the items can also be assigned features, which also serves as an alternative way of assigning cost factors. Additionally, a boolean field represents whether the item is active or it is not. Finally, an integer field stores the level in the network item hierarchy; which may be used for future developments.
- The `Node` class is a specialization of the `NetworkItem` class. It provides two additional fields to represent the node radius and the parent node in the hierarchy. So far, these fields may not be required. This class contains a static method to find a node within a set given its identifier.
- The `Link` class is a specialization of the `NetworkItem` class. It contains two fields which are references to the start and end node of the link. Additionally, an extra field indicates whether the link is bidirected or not. This class also contains a static method to find a link within a set given its numeric identifier.
- The `Feature` class is identified by a name. It contain a set of costs.
- The `Cost` class relates a cost factor to an specified situation.

5.3.2 `EditionKnowledge` Subsystem

Figure 5.2 displays the class diagram for the classes in the `EditionKnowledge` subsystem. This section describes each of these classes in higher detail.

- The `VisualNetwork` class serves as an edition wrapper for the `Network` class in the `ModelKnowledge` subsystem. This class also contains a set of planes and a set of visual links.
- The `Plane` class represents a bidimensional entity which is loaded into the edition tool. A plane its identified by its name, and contains a background image, so that visual nodes are placed over this image. It must be realized that while nodes must belong to a plane, links may or may not belong to a single plane, as a link can join

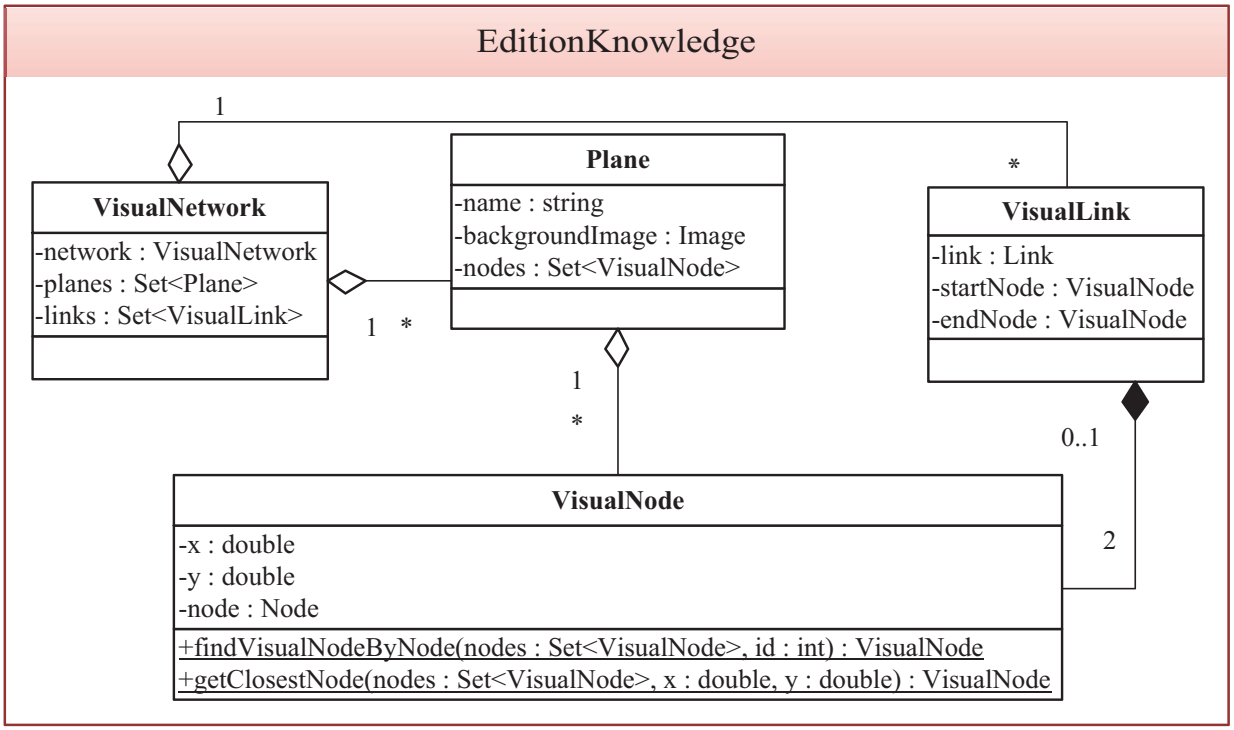


Figure 5.2: Classes for the `EditionKnowledge` subsystem

two nodes in different planes. This explains why the `VisualNetwork` class contains a set of visual links, while the `Plane` class contains a set of visual nodes.

- The `VisualNode` class serves as the graphical representation of a node. It contains two position attributes, x and y, which represents the visual node position over the plane. Moreover, this class also contains an attribute which references to a node in the `ModelKnowledge` subsystem. This class presents several static methods, which may be useful for some purposes. The first method allow to find a visual node in a set given a node from the *ModelKnowledge* subsystem. Another static method allows to find the visual node in a set which is closest to a given coordinates.

- The `VisualLink` class serves as a graphical representation of a link. It contains two attributes which reference the start and end visual nodes, as well as another attribute to refer the actual link in the `ModelKnowledge` subsystem.

5.4 Functional Architecture: Logic Tier

This section will detail the different classes of the logic tier. Moreover, it will discuss the choice of an algorithm over others when it is required.

5.4.1 `DataGateway` Subsystem

Figure 5.3 displays the class diagram for the classes within the `DataGateway` subsystem. This section describes each of these classes in higher detail.

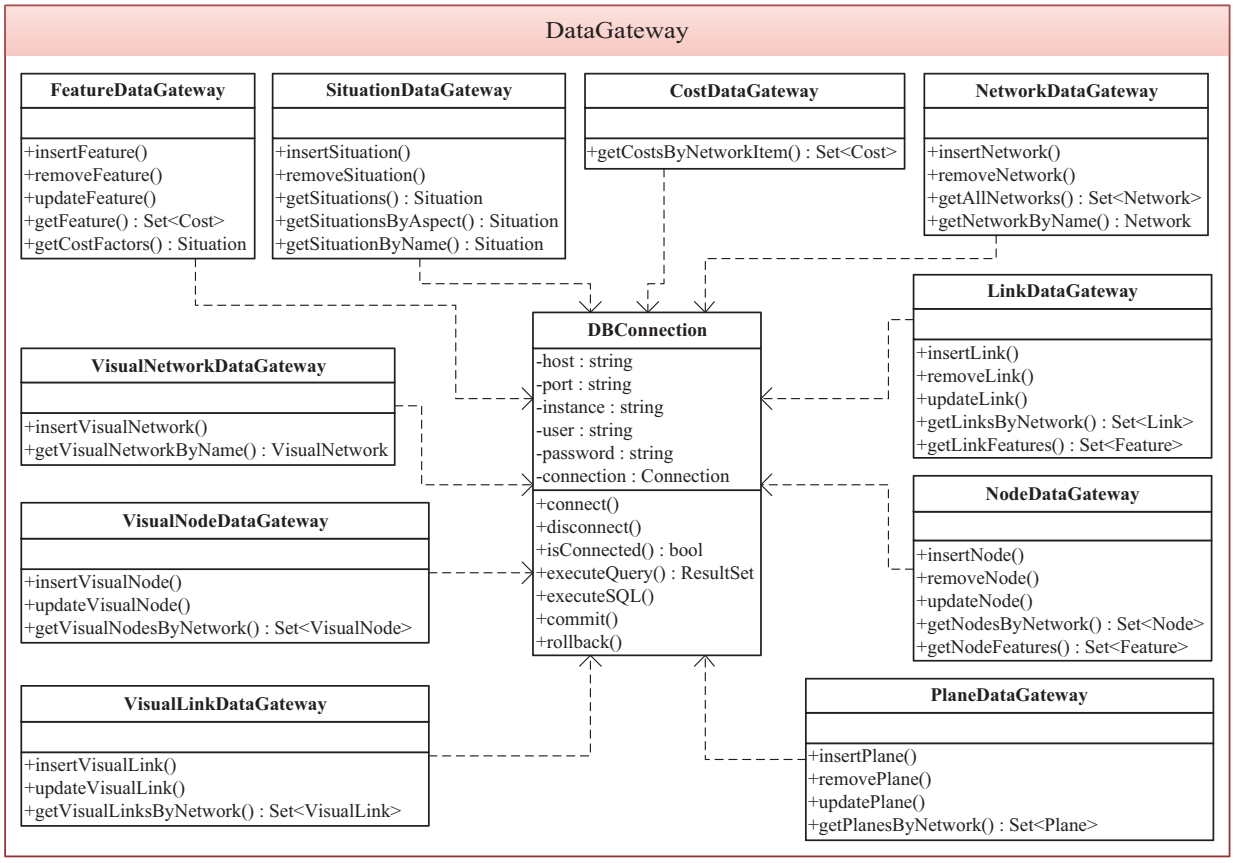


Figure 5.3: Classes for the `DataGateway` subsystem

5.4.1.1 DBConnection

The `DBConnection` class allows to execute basic operations over the database. It uses the *Oracle Database 11g JDBC* driver library, which provides Java classes to establish a communication with an Oracle database. This class provides methods to:

- start a connection with the database, given the information required for the connection to be established.
- close an open connection to the database.
- inform whether a connection is established or not.

- execute a query over the database and return a set of results.
- execute any other SQL instruction, such as insertions, deletions or updates.
- commit or rollback the changes, if applicable.

All the methods regarding the execution of SQL instructions must be designed to prevent SQL injections. This can be easily achieved using the tools provided by the *Oracle Database JDBC* library.

The application may take advantage of the use of a transactional database management system in order to allow transactions for the network edition process. Thus, the user is given the option to commit or rollback the changes at any time.

5.4.1.2 DataGateway Classes

DataGateway classes provides basic CRUD functionality for each of the classes in the data tier. The next methods are provided by these classes:

- Insert a new object in the database.
- Remove an existing object from the database. This function is not available for some objects which are deleted *on cascade*.
- Update an existing object in the database. This function is not provided for some objects which cannot be updated.
- Select an object or a set of objects from the database. This selection may be filtered by some fields (to return some results), filtered by an identifier (to return only one result) or not filtered (to return all the results).

5.4.2 `Edition` Subsystem

The `Edition` subsystem is composed by only one class containing the functions required by the edition functionality, and an auxiliar class. Actually, most of this functionality is provided by the `DataGateway` subsystem, as well as the classes in the presentation tier. It must be taken into account that the use of the interface for edition purposes (all the components of the GUI belongs to the presentation tier), carries out in most of the cases direct modifications over the database. Due to this fact, most of the times direct communication is performed between the `UI_Edition` subsystem and the corresponding data gateway. Figure 5.4 shows the class diagram for the `Edition` subsystem.

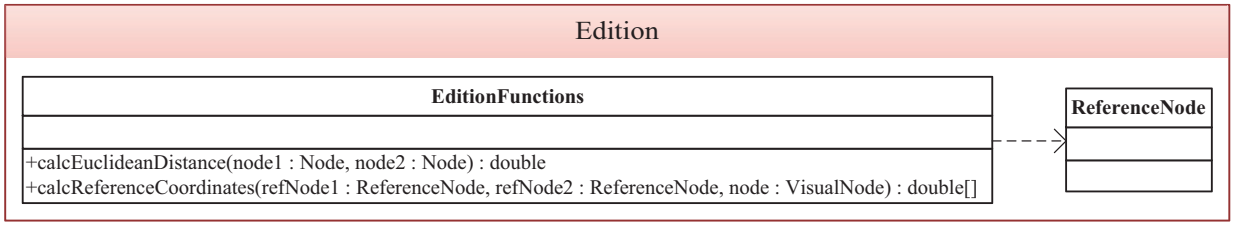


Figure 5.4: Classes for the `Edition` subsystem

5.4.2.1 ReferenceNode

The `ReferenceNode` class is a subclass of `VisualNode`. Actually, it does not provide any additional field or method to this class; however, it has a different graphical representation. Reference nodes are used for calculating the coordinates of other nodes in the network. The procedure for this operation is explained in the description of the `EditionFunctions` class.

5.4.2.2 EditionFunctions

This class only contains two auxiliary methods. The first one stands for calculating the euclidean distance between two nodes. Given the coordinates of the first node (x_1 and y_1) and the coordinates of the second node (x_2 and y_2), the euclidean distance is defined as:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

For our application, the euclidean distance is calculated in order to set an initial basic cost to a link. By default, the cost for a node is 0, while the cost for a link is the euclidean distance between his initial and ending nodes.

The second method calculates the position of one node given the position of two different nodes. The purpose of this operation is to ease the process of assigning coordinates, as now the user only has to define two reference nodes in the network and assign coordinates to both of them. Each reference node contains both real coordinates (x_i and y_i, where $i \in \{1, 2\}$) and graphical coordinates over the edition plane (x_{vi} and y_{vi} where $i \in \{1, 2\}$). Given that the two reference nodes are located in different places and the segment they generate is not contained neither in the x axis nor in the y axis (i.e., $x_1 \neq x_2$, $y_1 \neq y_2$, $x_{v1} \neq x_{v2}$ and $y_{v1} \neq y_{v2}$), the scale between the visual plane and the real network can be calculated (for two dimensions) as follows:

$$scale_x = \frac{x_2 - x_1}{x_{v2} - x_{v1}}$$
$$scale_y = \frac{y_2 - y_1}{y_{v2} - y_{v1}}$$

Given a node n for which we know its visual coordinates (x_{vn} and y_{vn}) but we have no information about its real coordinates (x_n and y_n), these can be calculated given the next formula:

$$x_n = x_1 + (x_{v1} - x_{vn}) \times scale_x$$

$$y_n = y_1 + (y_{v1} - y_{vn}) \times scale_y$$

Then, we can take advantage of the two reference nodes to calculate the coordinates of any other node in the network.

5.4.3 `Simulation` Subsystem

The `Simulation` subsystem, which contains the logic to run a simulation over the situation model, is formed by the classes which are shown in figure 5.5.

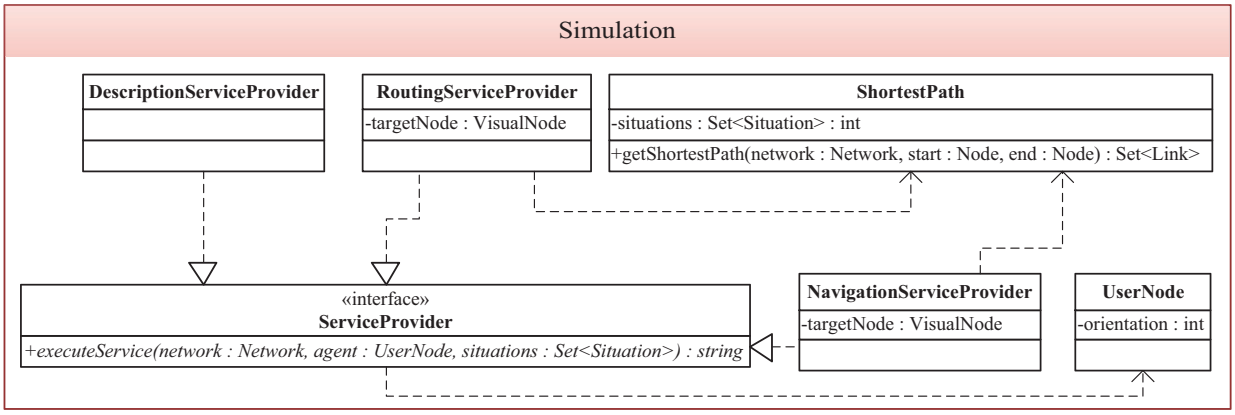


Figure 5.5: Classes for the `Simulation` subsystem

5.4.3.1 UserNode

The `UserNode` class is a subclass of `VisualNode`, and represents the user (or agent) location in the network. Apart from the fields of the `VisualNode` class, it also includes a field to represent the agent orientation. This orientation may be required by some services, such as the navigation service, to give the most suitable response (for example, the orientation must be known to tell the agent whether he has to turn to a different direction in order to achieve his goal).

5.4.3.2 ShortestPath

The `ShortestPath` class is used by the routing and navigation services to calculate the shortest path between the user location and its desired goal. The method `getShortestPath()` receives a network and source and target nodes. Moreover, a class field contains a set of applying situations, which is used to calculate the cost factors for each network item.

Given that the network is represented as a directed weighted graph, we can study different algorithms to find the shortest path between two nodes.

A first approach is found in the classic Dijkstra's algorithm [Dijkstra, 1959]. This algorithm is complete (it finds a solution) and admissible (it provides the optimal solution). Thus, it may be a good solution for our problem. However, it can be really inefficient for big networks, for which further approaches should be observed.

The use of heuristic search techniques can improve the performance, as the search is no longer blind and instead is guided by a heuristic function. For instance, the A* (A-Star) algorithm could be used [Hart et al., 1968]. This algorithm is complete, and it is also admissible as long as the heuristic function is admissible as well (for an heuristic function to be admissible, it must never overestimate the real cost).

Although A* would lead to a performance gain over Dijkstra, both in the temporal and spatial aspects of the computational complexity, in order to obtain the shortest path we need to design an admissible heuristic, and that will not be always possible. A good approach for an admissible heuristic would be the euclidean distance between the source and the target nodes. However, although this may be admissible most of the times, it could overestimate the real cost if the user have edited the links costs so that they are smaller than the euclidean distance, or if there are cost factors which are smaller than 1.

Despite that A* would improve the performance, it would still require exponential space to operate. On the other hand, there are some alternatives which are heuristic guided and can also provide an optimal solution if the heuristic is admissible, such as IDA* (Iterative Deepening A*) [Korf, 1985]. This algorithm would require more expansions than A*, and therefore may take slightly more time to find a solution; but it will require much less memory.

While the previous algorithms are intended to calculate the optimal solution, it may not be required. Given that time is an important constraint, in the case of big networks a greedy algorithm may be used, which may not provide an optimal solution but a suboptimal one instead. An example of these algorithms are hill climbing (which takes the most promising node in each step but considering only ony path, which most of the times will lead to a local optimum) and beam search, which behaves like hill climbing but considering more than one path at a time.

In fact, an hybrid solution may provide a good behavior for our system. While Dijkstra or A* will provide good solutions for small networks, greedy algorithms may be used for bigger networks. In this case, a greedy algorithm can be used and, when the user is closer

to the target node, it is substituted by an admissible algorithm.

Finally, some non-conventional techniques shall also be taken into account, such as Ant Colony Optimization (ACO) techniques. Some advances in ACO are promising regarding hugh dynamic networks [Rivero, 2012], such as the ones that may be used within this project, where the situations may change dynamically the costs of the different paths; thus providing a convenient and efficient way to constantly update the route to adapt the new situations.

For this project, Dijkstra's algorithm will be used for calculating the optimal path. Although more interesting approaches (in terms of efficiency) have been observed, Dijkstra is the only one which guarantees optimality given that no heuristic functions can be implemented. It must be noticed that optimality was required in requirement FR-SI-11 (minimum cost path), which is described in table 3.44. However, the study of the implementation of other algorithms is proposed as a future work.

It must be noticed that performance can also be improved if the system is not required to calculate the path every time the user moves to a new node. For these cases, the system may check whether the new node is included in the previously calculated path and take advantage of this to save computational effort.

5.4.3.3 ServiceProvider

The `ServiceProvider` interface in motivated as new services can be easily integrated into the system, as long as they comply with the interface. This interface contains only the method `executeService()`, which must provide all the logic for the service execution and returns its output.

So far, three classes implementing the `ServiceProvider` interface will provide the required services: the `DescriptionServiceProvider` class provides the description of the user location, the `RoutingServiceProvider` and `NavigationServiceProvider` classes obtain the shortest path between the user location and its desired goal. While the routing service prints all the route to the goal, the navigation service provides the instructions one by one while the user is moving across the network. These classes require to store the target node in order to calculate the path.

The `DescriptionService` class provides the description of the node which is closest to the user location.

The `NavigationService` class generates an instruction which the user must follow to go a step closer to his desired goal. This instruction tells the user whether he must either turn to a different direction, go back or go straight, given its current position and orientation. Moreover, the system provides the description of the next node to be visited, if it is available, in order to provide the user with helpful and reliable instructions.

Figure 5.6 shows a cartesian plane representing a possible distribution of the user, the

source node (which is shown translucent in the figure) and the target node, which in the figure is shown with an incoming link.

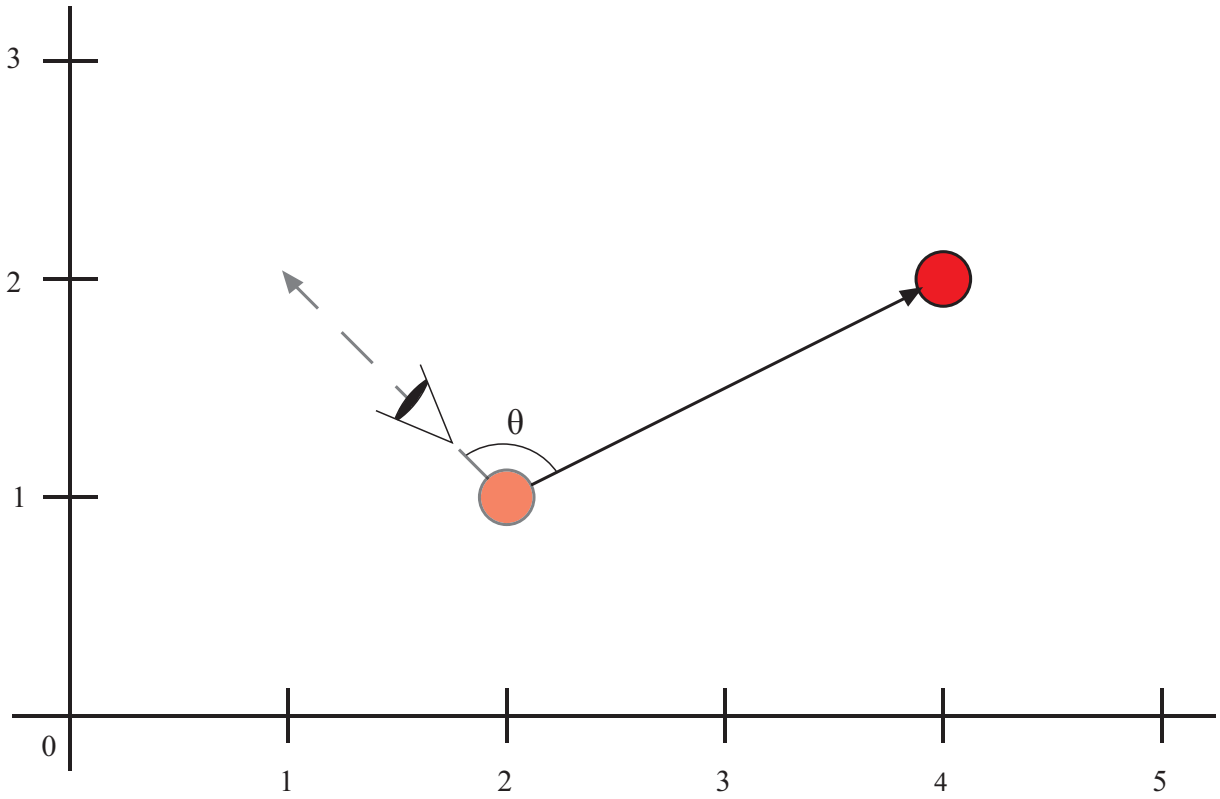


Figure 5.6: Geometric state of the plane

In order to get the next instruction, the system must calculate the angle θ, which is shown in the figure. Actually, this angle can be calculated using the dot product of the two vectors:

$$\overrightarrow{v_{target}} = \begin{pmatrix} 2 & 1 \end{pmatrix}$$
$$\overrightarrow{v_{user}} = \begin{pmatrix} -1 & 1 \end{pmatrix}$$

We can calculate the angle θ using the dot product and the vector lengths:

$$\arccos\theta = \frac{\overrightarrow{v_{target}} \cdot \overrightarrow{v_{user}}}{||\overrightarrow{v_{target}}|| \cdot ||\overrightarrow{v_{user}}||}$$

In our sample, $\theta \approx 113°$.

Moreover, not only the angle is required, but also the angle direction, in order to know whether the user must turn left or right. The angle direction can be calculated using the cross product between the two vectors:

$$\overrightarrow{v_{user}} \times \overrightarrow{v_{target}} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ \overrightarrow{v^1_{user}} & \overrightarrow{v^2_{user}} & 0 \\ \overrightarrow{v^1_{target}} & \overrightarrow{v^2_{target}} & 0 \end{vmatrix}$$

The result of the cross product will be a vector, and the sign of the coefficient of the component of the third dimension will determine the angle direction. The sign of the result will be added to the angle to indicate its direction. In our example, the value of the cross product is

$$\overrightarrow{v_{user}} \times \overrightarrow{v_{target}} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ -1 & 1 & 0 \\ 2 & 1 & 0 \end{vmatrix} = -3\vec{k}$$

where $\vec{k}$ represents the direction of the third dimension; and so the angle $\theta \approx -113°$.

So far, the system is able to provide six different instructions, each one for a given interval of angles. These instructions are shown in table 5.1. It can be seen that for the previous example, the system provides the instruction to turn right in order to achieve the next node.

Angle	**Instruction**
$-30°$ to $30°$	Go straight
$30°$ to $60°$	Turn slightly left
$60°$ to $120°$	Turn left
$-30°$ to $-60°$	Turn slightly right
$-60°$ to $-120°$	Turn right
$> 120°$ or $< -120°$	Turn back

Table 5.1: Routing instructions

The `RoutingServices` class provides a service quite similar to the navigation service. The only difference is that, while the navigation service only inform the user about the next step he must take to achieve its desired goal, the routing service provides a list of instructions to eventually achieve the final goal.

5.5 Functional Architecture: Presentation Tier

For this tier, we have decided to implement one class for each screen in the GUI. Although some additional classes and external libraries are required, these will not be detailed here, as most of them act as black boxes and serves for specific purposes.

Figure 5.7 shows the classes for the presentation tier. The attributes for the graphical components of these classes, which are automatically generated by the *NetBeans* GUI builder, are not shown in the figure.

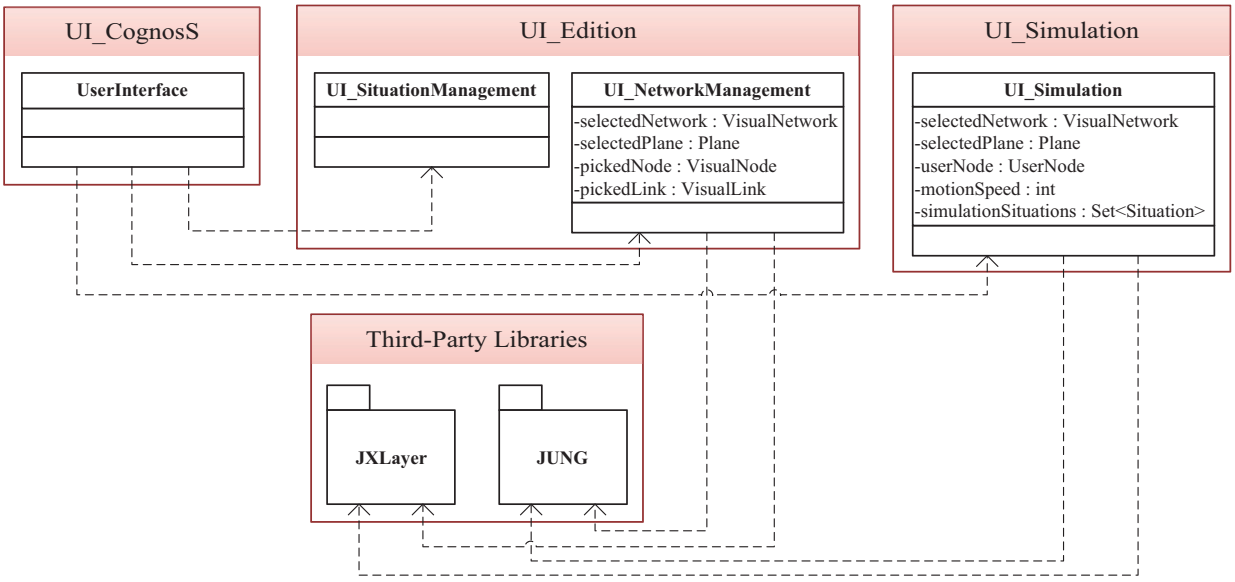


Figure 5.7: Classes for the presentation tier

The description of the required graphical components was already included in section 4.2.3. The purpose of this section is to detail the logical components of the different classes, as well as to introduce the external libraries used.

5.5.1 `UI_CognosS` Subsystem

The `UI_CognosS` subsystem only contains the `UserInterface` class, which shows the initial screen with the database connection form and the *about* panel. This class does not require any attribute apart from all the graphical components.

5.5.2 `UI_Edition` Subsystem

The `UI_Edition` subsystem contains two classes, one class (`SituationManagement`) contains the screen for situations management, while the other one (`NetworkManagement`) contains the screen for network edition.

The `SituationManagement` class requires some attributes which will store the currently selected network and plane, as well as the selected node or link, if any. It is required to store these items in order to enable their edition.

5.5.3 `UI_Simulation` Subsystem

The `UI_Simulation` subsystem contains the `UI_Simulation` class, which contains the screen for performing a simulation over a network. This class requires, apart from the attributes storing the graphical components, some additional attributes to work properly. First, the selected network and plane must be stored, as it contains the set of nodes and links for the simulation. The user node must also be stored, and its purpose is to keep the agent position and orientation over the network. The motion speed attribute is included to determine the speed by which the agent moves across the map, although it is not required. Finally, the set of selected simulations is stored and is passed to the service providers when a service is requested.

5.5.4 External Libraries

For dealing with some advanced features of the graphical interface, a couple of third-party libraries are used in this project. These libraries are briefed in this section.

5.5.4.1 JXLayer

The JXLayer library provides some useful functionalities for graphical interfaces. For example, the *JXLayer* class acts a `JPanel` which can be zoomed or rotated. This project takes advantage of this class to be able to provide a zooming feature to change the size of the loaded plane. Downloads for this library can be found in `http://jxlayer.java.net`.

5.5.4.2 Java Universal Network/Graph Framework (JUNG)

The JUNG library provides many interesting tools for dealing with graphs. Its functionality goes from rendering and drawing the graph to executing algorithms over it.

In this work, the renderers of the JUNG library are used to draw the graph. Many of the library classes will require to be overriden to change the default functionality in order to alter the nodes and links appearance, to print labels, etc. Moreover, the library also provides the Dijkstra algorithm to find the shortest path between two nodes of the graph. This algorithm can be used for the project, unless more sophisticated heuristic algorithms are implemented later.

Downloads, documentation and examples for this library can be found in `http://jung.sourceforge.net`.

5.6 Data Storage

This section describes the implementation of the data stores of the project. As it was stated in section 4.3, two independent data stores are required, the first one containing the model knowledge base and the second one storing the edition knowledge database.

5.6.1 Model Knowledge Base

Figure 5.8 shows the relational graph [Smith, 1985] for implementing the model knowledge base. This graph provides a detailed description of the database architecture, which may be used for modeling the database using the SQL language.

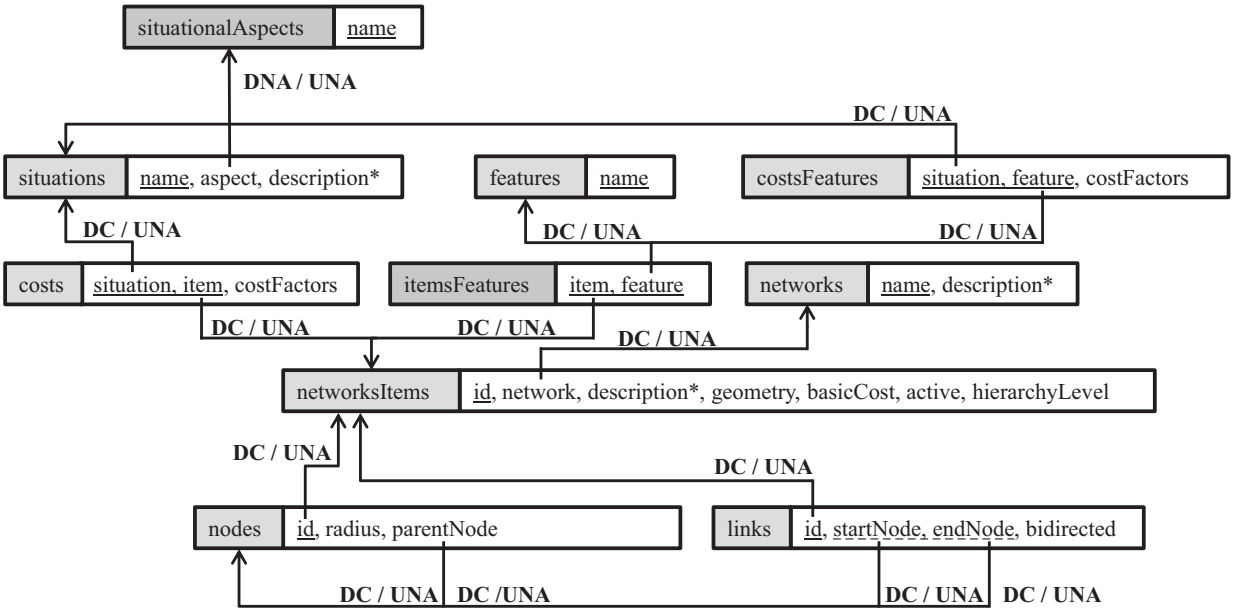


Figure 5.8: Relational graph for the model knowledge

The relational graph does not show the field types, although it can be deduced from the data types shown in the class diagrams from the previous sections. It may be noticed that all the referential integrity keys are *On Delete Cascade, On Update No Action*, but for the foreign key from the `situations` relation to the `situationalAspects` relation, which is *On Delete No action, On Update No Action*. Actually, this last relation is intended to be a validation table, where its values are fixed and are not supposed to be deleted. This fact may also require a mechanism, such as a trigger, to prevent modifications in this table.

The rule for deletions allows the removal of elements on cascade, which makes sense for all the cases in our design. *No action* is considered for updates, as in the futures the names will refer to concepts in an Ontology, and it is not desired that an editor alters the semantical value introduced by another editor.

Regarding additional semantics not shown in the graph, the cost factors must be greater or equal to zero, as negative cost factors would probably make it impossible to calculate the shortest path between two nodes. These semantics can be implemented by means of a table constraint. Moreover, the network items numeric identifier will be autoincremental. This mechanism can be implemented through a sequence and a trigger.

5.6.2 Edition Knowledge Base

Figure 5.9 shows the relational graph for the edition knowledge base. This relational graph duplicates the structure of the model knowledge, and includes the new relations for supporting the edition operations. The need for the replication of the model knowledge was justified in section 4.3.

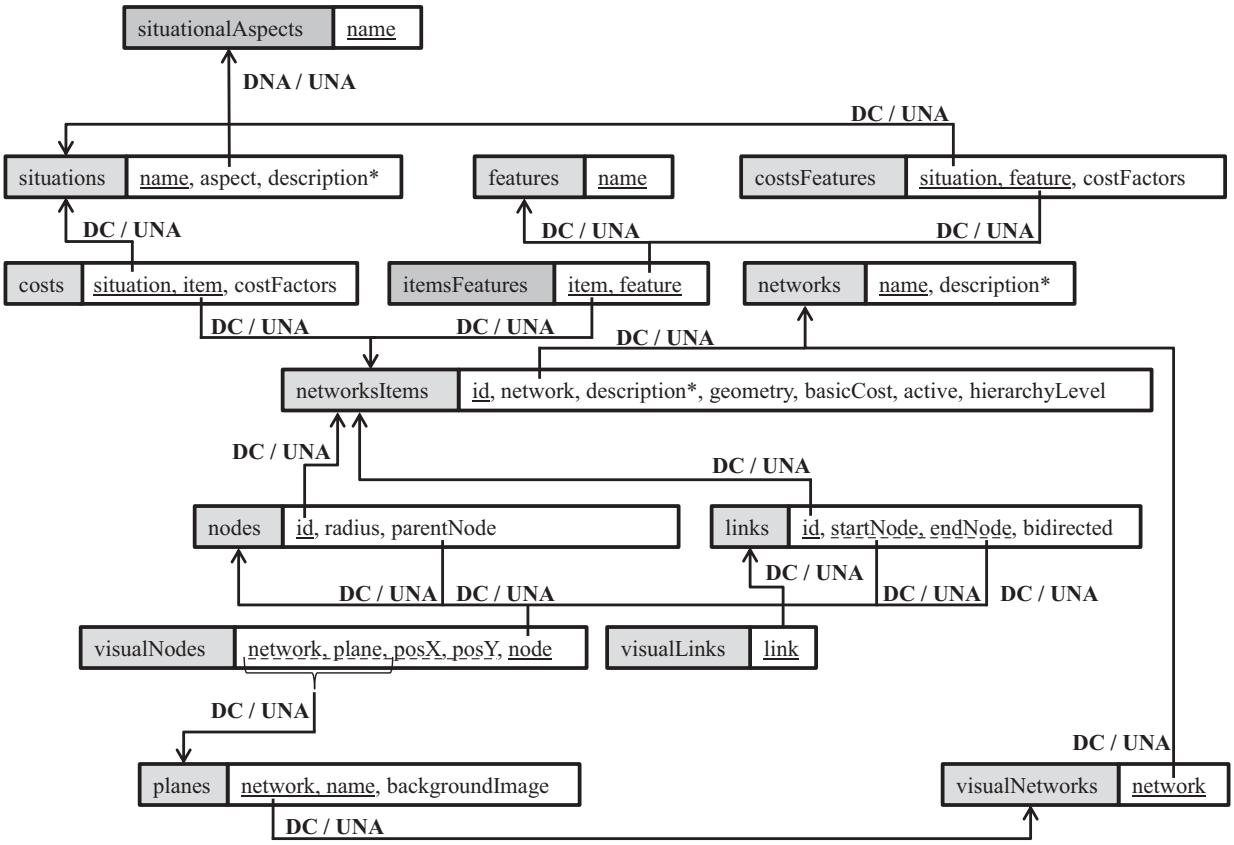


Figure 5.9: Relational graph for the edition knowledge

The redundancy which was mentioned in section 4.3 still remains in the database implementation. For this reason, we will implement also a trigger which will prevent possible inconsistencies in the data. This trigger will ensure that, when a visual node is inserted in the database, it belongs to the same network that its associated real node. Moreover, the application will be developed in order to prevent this inconsistency.

This page has been intentionally left blank.

Chapter 6

Validation and Evaluation

The main purpose of this chapter is to check that the system successfully does what it is required to do, given the software requirements specified in section 3.2. This phase is called *validation* and it is essential to guarantee the quality of a software product, and to accept it. Section 6.1 shows the results for the validation tests. In the cases where a test reveals that some of the requirements cannot be validated, corrective actions will be proposed to solve this fact.

Additionally, this chapter proposes metrics for a system evaluation in section 6.2. The proposed evaluation is a comparative one, and given that there are no other systems of similar purposes to which this work could be compared, the tool is finally compared over the manual knowledge edition. This evaluation will consider both a quantitative and a qualitative approach by means of measurements of quality and performance, as well as a subjective questionnaire; finally concluding whether the developed tool presents advantages over the direct edition of the model knowledge.

6.1 Validation

The process of software validation will check that the requirements specified by the user are covered by the system. To do so, the validation tests from section 3.5 are executed. In order to perform this execution correctly, it must checked that the postconditions are met given that the test steps are executed over the system which already satisfied the preconditions. If that happens, then the test is successful and the requirements covered by the validation test are satisfied, and this fact is indicated with a green checkmark. Otherwise, it is important to analyze why the test failed and to propose a corrective action. In this case, the requirements which are not validated can be essential or not, and the failure is indicated with a red or orange cross respectively. Tables 6.1 to 6.12 details the results for all the validation tests.

Validation Test	Test 01 (table 3.49).
Test Name	Database connection.
Result	✓ Ok.
Corrective action	*Not required.*

Table 6.1: Results for validation test 01 (Database connection)

Validation Test	Test 02 (table 3.50).
Test Name	Situations management.
Result	✓ Ok.
Corrective action	*Not required.*

Table 6.2: Results for validation test 02 (Situations management)

Validation Test	Test 03 (table 3.51).
Test Name	Features management.
Result	✓ Ok.
Corrective action	*Not required.*

Table 6.3: Results for validation test 03 (Features management)

Validation Test	Test 04 (table 3.52).
Test Name	Network selection.
Result	✓ Ok.
Corrective action	*Not required.*

Table 6.4: Results for validation test 04 (Network selection)

Validation Test	Test 05 (table 3.53).
Test Name	Network selection.
Result	✗ Wrong (not essential). Planes management is not implemented.
Corrective action	Due to its cost, planes management feature has not been implemented. So far, only one default plane per network its supported. For this reason, the requirements FR-MA-13, FR-MA-14 and FR-MA-15 are not validated. However, all these requirements have their priority defined as *optional*, and they are not essential. The support for planes management is included as a future line for the short term.

Table 6.5: Results for validation test 05 (Plane selection)

Validation Test	Test 06 (table 3.54).
Test Name	Plane visualization.
Result	✓ Ok.
Corrective action	*Not required.*

Table 6.6: Results for validation test 06 (Plane visualization)

Validation Test	Test 07 (table 3.55).
Test Name	Network items management
Result	✗ Wrong. The test does not pass because cost factors cannot be assigned to a network item.
Corrective action	Network items can be edited to update their description, coordinates, basic cost, activeness and features. However, they cannot be assigned cost factors directly, and for this reason requirement FR-MA-27 cannot be validated. So far, cost factors can be assigned indirectly by means of features. As this requirement is defined as *conditional*, the support for the assignment of cost factors is included as a future line with the highest priority.

Table 6.7: Results for validation test 07 (Network items management)

Validation Test	Test 08 (table 3.56).
Test Name	Agent simulation.
Result	✓ Ok.
Corrective action	*Not required.*

Table 6.8: Results for validation test 08 (Agent simulation)

Validation Test	Test 09 (table 3.57).
Test Name	Simulation services.
Result	✓ Ok.
Corrective action	*Not required.*

Table 6.9: Results for validation test 09 (Simulation services)

Validation Test	Test 10 (table 3.58).
Test Name	Routing and navigation services.
Result	✓ Ok.
Corrective action	*Not required.*

Table 6.10: Results for validation test 10 (Routing and navigation services)

Validation Test	Test 11 (table 3.59).
Test Name	Big networks
Result	✓ Ok.
Corrective action	*Not required.*

Table 6.11: Results for validation test 11 (Big networks)

Validation Test	Test 12 (table 3.60).
Test Name	Interoperability.
Result	✓ Ok.
Corrective action	*Not required.*

Table 6.12: Results for validation test 12 (Interoperability)

In conclusion, requirements FR-MA-13, FR-MA-14, FR-MA-15 and FR-MA-27 are not validated. Any of these are defined as *essential*, so their validation is not absolutely required for the system validation. However, in all the cases, a corrective action is proposed and the validation of all the requirements is left as future lines with the maximum priority.

6.2 Evaluation

A good system evaluation would be able to indicate whether the system successes in its purposes, and whether it provides some advantages over other systems.

So far, no similar systems with the same purpose were found, so this kind of evaluation cannot be performed. However, for the edition process, a comparative evaluation could be achieved in order to observe whether the tool provides some advantages over direct edition of the model knowledge.

For this purpose, an experiment has been designed in order to model a given network by two different ways. The first one will edit the model directly over the model knowledge base, using some auxiliary procedures developed in order to ease this task (by avoding the use of specific SQL commands). The second way will edit the model by means of the management tool developed in this project. Figure 6.1 shows the instructions sheet which was distributed among the subjects, which details the tasks which have to be executed in the experiment.

This experiment will enable to gather both qualitative and quantitative information:

- Through direct observation, the experiment manager will be able to measure the times for each of the tasks. It is expected that the edition tool increases the performance of the operation.
- After the tasks are completed, the experiment manager will be able to measure the accuracy of both systems by observing the deviation of each node to its original location.

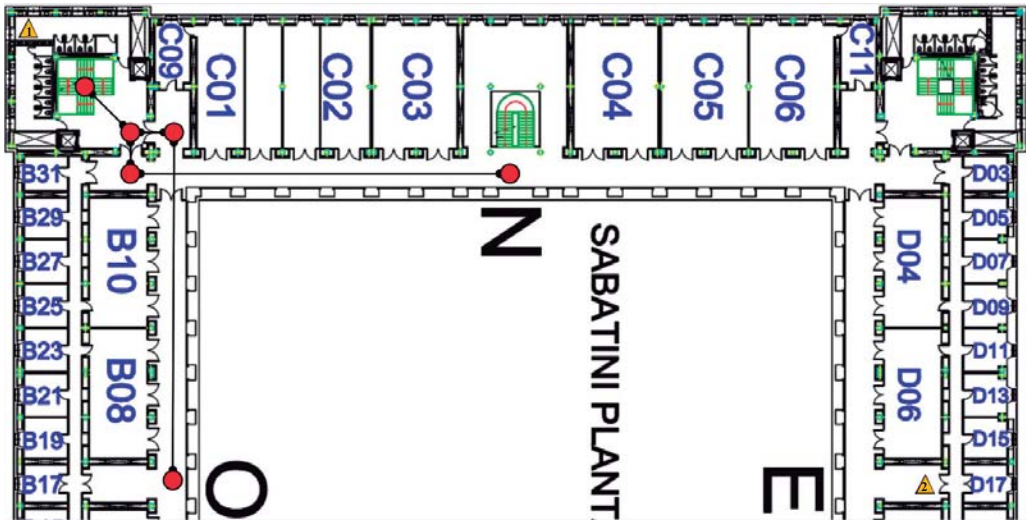
Cognos.S

Circumstantial Knowledge Management

Instructions

Please, start by reading the instructions carefully.

In order to evaluate the developed management tool, you will be required to edit the next network for a situation model:



Reference points △1 and △2 are physically located in the coordinates (0, 0) and (100, 50). The coordinates for each node must be extracted from this information, as checking the accuracy is part of the evaluation.

You are asked to perform the next tasks:

A. <u>Model the network directly over the model knowledge base.</u>

For this purpose, you will be provided with access to a database and will be able to execute SQL instructions. You must follow the next steps:

1. Establish a connection to the database (the access information is attached in a different document).
2. Create a new network in the table *Networks*, named "Ev-*your_name*-Man". You can use the next procedure:
   `insert_network(`**`network_name`**`) returns` **`network_name`**
3. Model the network provided in the figure. The next procedures are provided to ease this task:
   `insert_node(`**`network_name`**`,` **`x`**`,` **`y`**`) returns` **`node_id`**
   `insert_link(`**`network_name`**`,` **`node1_id`**`,` **`node2_id`**`) returns` **`link_id`**

B. <u>Model the network using the management tool.</u>

For this purpose, you will be given a copy of the edition tool, the plane background image and the user manual. You must follow the next steps:

1. Start the application and establish the connection to the default database.
2. Go to the *network management* tab, create a new network called "Ev-*your_name*-Tool". Load the network.
3. Select the provided image as background image.
4. Model the network provided in the figure. You can use the *reference nodes* feature to avoid calculating the position for each node. You can change the node scale if you wish.

After finishing with the two operations, please fill the attached survey.

Figure 6.1: Instructions for the experiment

- By filling a survey, the experiment could also gather some subjective information such as whether the user considers that the system is comfortable, reliable, efficient and

intuitive. The survey also asks for the advantages and disadvantages of each of these systems and allows the user to express other considerations. Figure 6.2 shows the questionnaire that was distributed among the subjects, and which they filled after completing the two tasks.

Cognos.S

Circumstantial Knowledge Management

Survey

Please, take some minutes to fill this survey after performing the two tasks.

Survey ID 0

Name

1. Evaluate the next statements in a scale from 1 to 5 (being 1 *strongly disagree* and 5 *strongly agree*) for each of the two systems.

Question	System A	System B
I felt good when performing the task (the system is comfortable).	1☐ 2☐ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☐
I found the functionality which I required easily (the system is intuitive).	1☐ 2☐ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☐
I think I completed the task successfully and without errors (the system is reliable).	1☐ 2☐ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☐
I think the task did not take too much time to be completed (the system is agile).	1☐ 2☐ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☐

2. What do you think are the advantages of System A?

3. What do you think are the disadvantages of System A?

4. What do you think are the advantages of System B?

5. What do you think are the disadvantages of System B?

6. In what situations do you think System A would be convenient?

7. In what situations do you think System B would be convenient?

8. Please, express here any other observation.

Thank you for your time.

Figure 6.2: Questionnaire for the experiment

As the tool is aimed at a very specific audience, the population chosen for the evaluation is small: it is formed by six persons. For this reason, a test of significance will not be performed over the experiment. From the evaluation population, one half are experts in

the annotation of spatio-temporal knowledge and the other half are familiarized with the annotation of knowledge in other areas. The next academic profiles can be found in the subjects of the experiment: one mathematician, one linguist, one pharmacist and three computer engineers. The targeted subjects are in the age range from 25 years to 40 years.

Each person was given the instructions sheet, a copy of the survey and a copy of the edition tool user manual. During the experiment, the times were measured for each of the tasks, which were carried out in different orders by the experiment subjects. The next sections detail the results of the evaluation and elaborate some conclusions on them. Meanwhile, appendix C contains the raw data gathered in the experiment.

6.2.1 Performance

Table 6.13 shows the average times and the standard deviation for running the modeling tasks using the direct model edition method and the tool. As it can be observed, the time is almost four times lower when the edition tool is used.

	Direct Edition	**Edition Tool**
Average	27' 20"	7' 21"
Std. Deviation	4' 43"	1' 33"

Table 6.13: Times for modeling tasks

6.2.2 Accuracy

The accuracy is checked by observing whether the nodes positions in the network modeled by the user correspond with the coordinates of the nodes from the original network. Specifically, the error is calculated as the sum of the Manhattan distances for each of the nodes. This means, if o is the original node and n^i $\forall i \in [1, 6]$ are the nodes placed by the user in the modeled network, the error is calculated using the next equation:

$$E = \sum_{i=1}^{6} \left(|n_x^i - o_x| + |n_y^i - o_y| \right)$$

Table 6.14 shows the average error (in meters) in the modeling process for each of the edition methods, as well as the standard deviation.

	Direct Edition	Edition Tool
Average	10.9019 meters	3.3970 meters
Std. Deviation	4.6152 meters	1.2877 meters

Table 6.14: Absolute error for modeling tasks

However, given that the total absolute error may not be significant, an alternative relative error can be considered. For instance, supposing that the technology of the physical layer used has a maximum margin of error of 1 meter, it will be considered 1 meter as the maximum acceptable error (100%). Then, the relative error can be calculated for each of the edition tasks using the next formula:

$$E = \frac{\sum_{i=1}^{6} \left(|n_x^i - o_x|\right) + \sum_{i=1}^{6} \left(|n_y^i - o_y|\right)}{12}$$

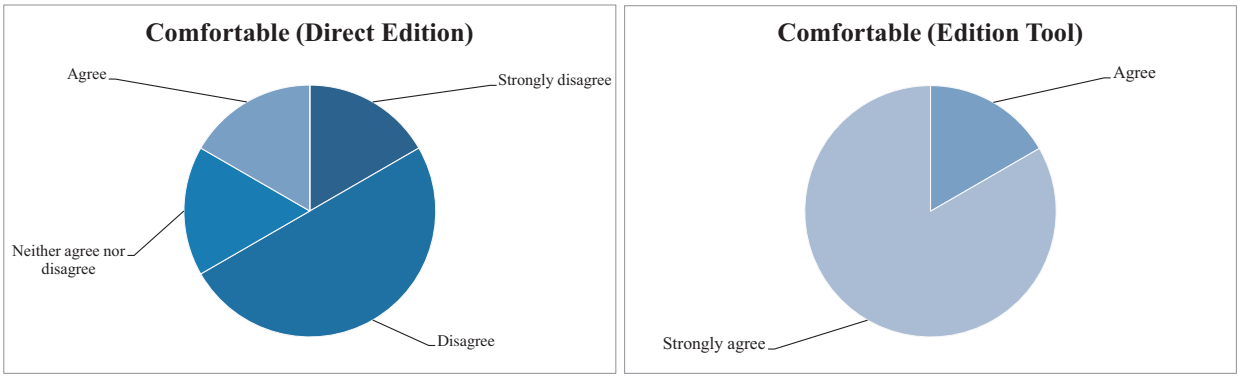
Finally, the results for the relative error are shown in table 6.15.

	Direct Edition	Edition Tool
Average	90.8495%	28.3089%
Std. Deviation	38.46%	10.73%

Table 6.15: Relative error (over 1 meter) for modeling tasks

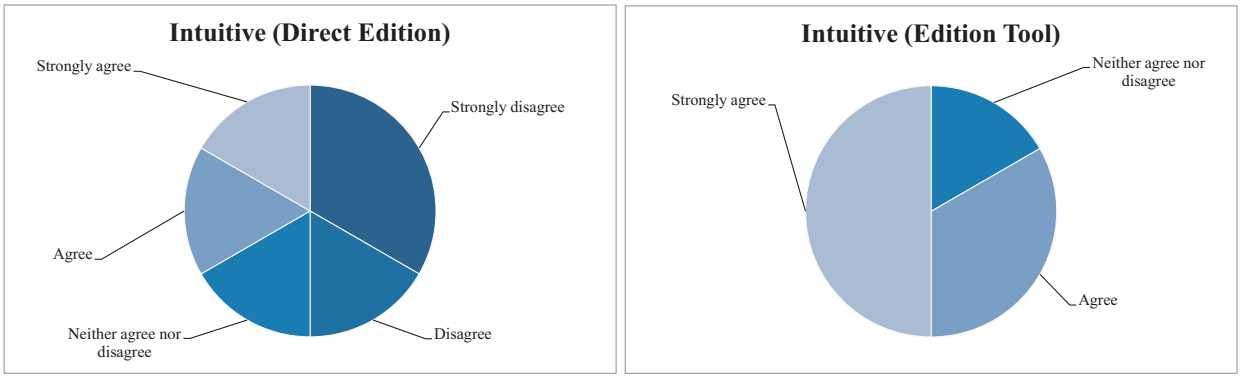
6.2.3 Subjective Evaluation

The experiment subjects were also asked for whether the system is comfortable, intuitive, reliable and agile (these questions are shown in figure 6.2); and they had to answer in a scale from 1 to 5. Figures 6.3 to 6.6 graph the results for this subjective evaluation.



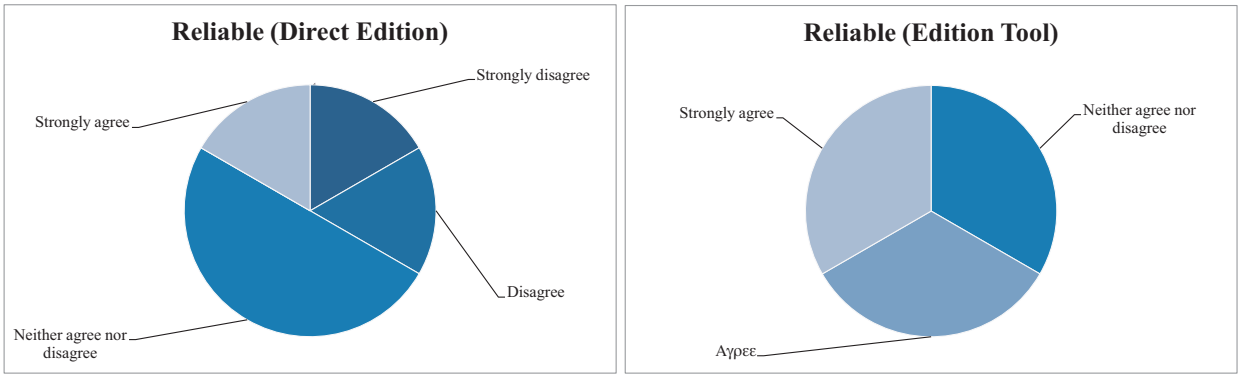
(a) Direct edition (b) Edition tool

Figure 6.3: Results for the subjective evaluation (comfortable)



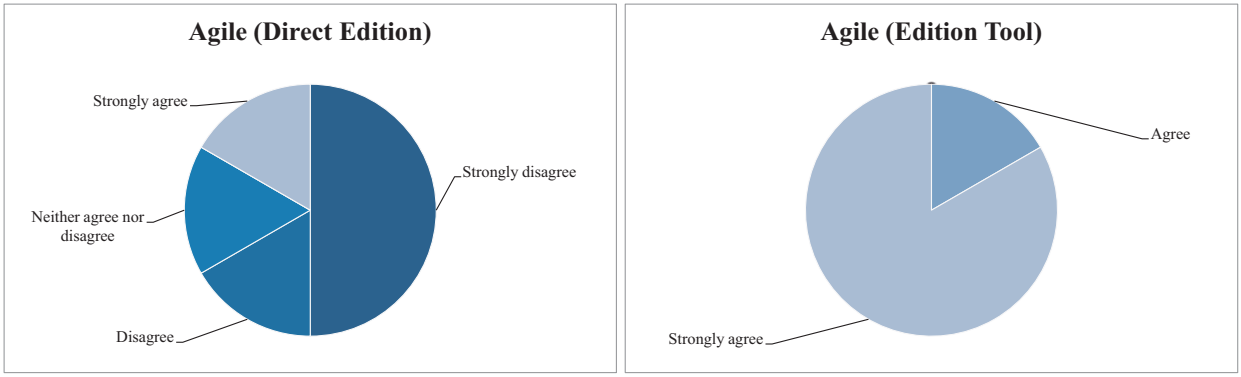
(a) Direct edition (b) Edition tool

Figure 6.4: Results for the subjective evaluation (intuitive)



(a) Direct edition (b) Edition tool

Figure 6.5: Results for the subjective evaluation (reliable)



(a) Direct edition (b) Edition tool

Figure 6.6: Results for the subjective evaluation (agile)

6.2.4 Conclusions

After the experiment is concluded, some conclusions are extracted regarding both the quantitative measurements performed during the evaluation (time and accuracy) and the sub-

jective evaluation completed by all the subjects.

In the first place, the information in table 6.13 shows that the tool improves the efficiency of the situation modeling task in a significant factor.

Regarding the quality of the annotations, by observing table 6.15, it can be concluded that the tool increases significantly the accuracy of the network modeling, as the relative error is remarkably lower when the edition tool was used.

The results in figures 6.3 to 6.6 clearly show that the experiment subjects prefer the tool edition over direct edition of the model knowledge. In average, users think that the edition tool is more comfortable (4.83 vs. 2.33), more intuitive (4.33 vs. 2.67), more reliable (2.83 vs. 4) and more agile (4.83 vs. 2.17).

Finally, some other conclusions can be extracted from subjective evaluation supported by the open questions in the survey:

- Direct edition of the model requires some technical skills or programming abilities, and thus the edition tool is more convenient for unexperienced users. However, some experienced users may prefer the direct edition as it allows to perform some scripting tasks.
- Although both systems require some training, using the edition tool is more intuitive as it is built in a common graphical style.
- The edition tool is more reliable as it shows the network graphically, whereas in the case of direct edition input errors are more difficult to detect.
- Modifications over the network are easier by using the edition tool.
- The edition tool does not enable the creation of a node directly by using its coordinates (it has to be created over the network using the mouse). This functionality has been proposed by expert users as a future line, so they can keep their current edition mechanisms and take advantage of the new functionalities and facilities.

Summarizing, the edition tool is considered more convenient and fits its purpose, as it enables fast knowledge edition and provides higher quality results with respect to the manual edition.

Chapter 7

Conclusions and Future Work

Finally, this last chapter presents the conclusions of the project. Actually, this project still has a long way to be improved. For this reason, the state of the project is revisited, and future work is proposed in this chapter as well. The purpose of this future work is to ensure the validation of all the user requirements in the first place, but also to include new functionalities in order to improve the user experience and to go a step further in the research in situation modeling.

7.1 Conclusions

When humans take part in an interaction, the surrounding context is influencing on many aspects of how this interaction is done. The context is determining our roles in the communication process, what we can say and what we cannot, and even what our gestures mean. The influence of the context is so strong that most of the times we do not realize about its existence. It has become a natural part of the interaction.

Unfortunately, this naturality is not extensible to computers. If we want a machine to be able of simulating a human in order to communicate with other humans, we must first identify all the components which take part in the human-human interaction, which is definitely not a simple task. Later, these components must be modeled, and its knowledge must be formalized. Again, this task is hard, but it is also essential in order to achieve natural interaction.

The purpose of this project was to design and implement a simple situation model, which formalizes and stores the knowledge of a subset of the material aspect of the context. Specifically, the model stores an spatial network, which is formalized as a graph. The graph formalization was chosen because it can be easily extended for observing the rest of the theoretical circumstantial aspects of the situation. Additionally, the implemented situation model supports the selection of situations, which may belong to any of the context aspects,

in order to provide partial support to these aspects.

Apart from the basic situation model, a management and simulation tool has been also implemented within this project. The edition functionality of the tool allows to edit the spatial network by adding, removing or updating its nodes and links; as well as it allows the management of context situations. When the edition tasks are completed, the tool allows to simulate the model's working to check the knowledge validity, and finally it allows to upload the result into a given knowledge base for a compatible situation model.

The simulation functionality of the tool runs over the edited network, which is stored in the situation model, and provides different services to the user, such as the description of his current position and the generation of the optimal route to his goal given the situations that apply during the simulation. A physical layer is emulated in order to determine the user location and its motion across the network.

After the development of the tool, an evaluation was carried out to compare the advantages of the tool over the manual edition of the situation model knowledge. The evaluation showed that the edition using the tool improved the performance and increased the accuracy of the process, and was also preferred by all the experiment subjects when they had to consider the comfort, intuitiveness, reliability and agility of the two methods. Moreover, the simulation functionality provides a way of checking the validity of annotations when uploading the knowledge into the base of the production system.

This work will be part of a research project in the LaBDA group, which will improve the situation model and the tool, will study the implementation of many of the future works described in the next section, and will integrate the model within the cognitive architecture for a natural interaction system presented in figure 1.1. A copy of the Cognos toolkit, including the edition tool developed within this project, is available in the next site: `http://labda.inf.uc3m.es/doku.php?id=es:labda_lineas:cognos`.

Moreover, comercial applications of the developed model can be found in the area of software for smartphones, where a location-aware system can be enriched with the situation model.

While most of the situation models existing in the state of the art are implemented for a specific system, the situation model of this project is generic and could be applied to any system. Moreover, the management tool allowing the knowledge edition of the situation model is a contribution to the state of the art, as no similar tools exist, even if the developed tool only supports the knowledge management over a specific situation model.

In order to work on this project, solid understanding on mathematics (graph theory, geometry, vectorial algebra), software engineering, object-oriented programming, graphical interfaces design and development, computer graphics, artificial intelligence (search algorithms, knowledge engineering) and databases was required. However, this is far from being as important as the knowledge obtained while working on the project. Not only an improvement of the expertise level on the previously mentioned areas, but also a better understanding on the components of the interaction between humans, and a comprehension

on the state of the art of related research areas. And again, this is still far from the fun of working on this project.

7.2 Future Work

The purpose of this section is to propose future lines of work for this project. The first aim of these works is to complete the project, in order to satisfy some user requirements which did not validate in the previous chapter. Moreover, this project is a first step for going into the research of situation modeling for a natural interaction system; and for this reason other future works are intended to improve the management tool and the situation model already implemented.

This section describes the future work for the project. Each line of work is defined in a table and contains a name, a description and a priority, which can be either *high*, *normal* or *low*, depending on whether the future work is intended to be carried out in the short term (within the next few months), the mid term (to be completed in the next year) or the long term respectively.

Regarding the taxonomy for the future works, they will be classified attending to whether they are related to the edition tool, to the situation model, to the Cognos toolkit or to the natural interaction system.

Actually, this work will be applied in a research project funded by the Spanish Ministry of Industry (CADOOH, TSI-020302-2011-21), and some of the future works will be taken into practice in the next few months.

7.2.1 Management Tool

Future works in this section propose some improvements on the edition tool, in order to provide new functionalities or enhance the user experience. From all of those, future works FW-01 and FW-02 are essential in order to validate the user requirements.

ID	FW-01.
Name	Costs factors assignment.
Description	A direct way of assigning cost factors to network items must be implemented, so that features are not strictly required for this purpose. This future work has the highest priority, as the development of this functionality is required to validate requirement FR-MA-27.
Priority	High.

Table 7.1: Future work FW-01 (Costs factors assignment)

ID	FW-02.
Name	Planes support.
Description	So far, the management tool only supports one plane per network, although the situation model knowledge base actually has multiplane support. Support for several planes is required to validate requirements FR-MA-13, FR-MA-14 and FR-MA-15, and for this reason the future work is given the highest priority, even if these requirements are not defined as essential. The support for several planes requires a new design of the user interface which allows to work with three dimensions, rather than two, in a comfortable and usable way.
Priority	High.

Table 7.2: Future work FW-02 (Planes support)

ID	FW-03.
Name	Multiple selection for network items.
Description	So far, in order to edit several network items, the edition must be performed for each item one by one. An alternative method for items selection could be implemented, in order to be able to select several elements at once. For instance, the user could press the key *Ctrl* to select more than one item, or even a selection window could be included, where all the items within a region are selected.
Priority	Normal.

Table 7.3: Future work FW-03 (Multiple selection for network items)

ID	FW-04.
Name	XML schema.
Description	To increase the model portability and interoperability, an XML schema could be specified. The networks could be saved as XML documents, thus the applications would not require a connection to a database to operate with the situation model. In fact, two schemas could be defined: the first schema containing the model knowledge and the second one formalizing the edition knowledge.
Priority	Normal.

Table 7.4: Future work FW-04 (XML schema)

ID	FW-05.
Name	Node creation by coordinates.
Description	Even if the tool graphical capabilities ease the task of modeling the network by improving the performance and the accuracy of this process, in some cases it might be interesting to be able to add a new node directly by inputting its coordinates, instead of placing it over the plane.
Priority	Low.

Table 7.5: Future work FW-05 (Node creation by coordinates)

ID	FW-06.
Name	Support for dynamic agents.
Description	So far, the only supported dynamic agent is the user who moves across the network. The dynamism for other elements can be partially simulated by using situations, which may change the item status. However, native support for dynamic agents (such as other people) is proposed as a future work. This agents would be able to change their status on every dimension of the network, i.e., they would be able to show some kind of evolution.
Priority	Low.

Table 7.6: Future work FW-06 (Support for dynamic agents)

ID	FW-07.
Name	Undo and redo.
Description	The application could include a functionality to revert the last changes. So far, this operation is partially implemented by taking advantage of a transactional system, which allows to commit and rollback the changes. However, a more sophisticated system could be implemented using a stack of historic operations.
Priority	Low.

Table 7.7: Future work FW-07 (Undo and redo)

7.2.2 Situation Model

The future works proposed in this section have as main aim to improve the functionality of the situation model.

ID	FW-08.
Name	Native support for the temporal aspect.
Description	While so far the situation model only provides native support for the spatial aspect (through a three dimensional network), the temporal aspect is only supported by means of situations. Native support for this aspect could be implemented directly in the situation model, by using temporal triggers or cron jobs.
Priority	Low.

Table 7.8: Future work FW-08 (Native support for the temporal aspect)

ID	FW-09.
Name	Native support for all the context aspects.
Description	The situation model could be improve to natively support all the context aspects, instead of just the material. While these aspects are currently being supported by means of situations, native support would provide a better integration among them, thus increasing the power and reliability of the system, as well as going a step further in the research of situation modeling. The result would be a multidimensional network, where each aspect of the context would contribute by adding one or more dimensions to the graph. On the other hand, the implementation of this future work can turn out to be really hard, and for this reason this future work is proposed with the least priority.
Priority	Low.

Table 7.9: Future work FW-09 (Native support for all the context aspects)

ID	FW-10.
Name	Efficient algorithms for the shortest path.
Description	In order to calculate the shortest path, the system is currently using the Dijkstra algorithm. However, better approaches were proposed in section 5.4.3, such as heuristic search algorithms (A* and IDA*), greedy algorithms (hill climbing or beam search) or even Ant Colony Optimization (ACO) techniques. Different alternatives could be implemented in the mid term in order to evaluate their response time and their optimality.
Priority	Normal.

Table 7.10: Future work FW-10 (Efficient algorithms for shortest path)

ID	FW-11.
Name	Trace tool.
Description	In order to analyze the reasoning processes of the situation model, a trace tool could be implemented.
Priority	Low.

Table 7.11: Future work FW-11 (Trace tool)

7.2.3 Cognos Toolkit

The tool is now isolated, as it is the result of an independent development project, but it will be integrated into a major toolkit for the complete edition of interaction knowledge (Cognos) [Calle et al., 2010, Calle et al., 2011] in the short term. Future works proposed in this section have the purpose of integrating the situation model tool in the Cognos toolkit, thus enriching the other models.

ID	FW-12.
Name	Inlaying in Cognos toolkit.
Description	The edition tool will be incorporated to the Cognos toolkit, by making the application interface consistent with the already existing tools.
Priority	High.

Table 7.12: Future work FW-12 (Inlaying in Cognos toolkit)

ID	FW-13.
Name	Integration in Cognos knowledge base.
Description	The edition knowledge will be integrated in the Cognos knowledge base. When this integration is completed, networks will be associated to a corpus, and the user will be required to select a corpus in the management tool.
Priority	High.

Table 7.13: Future work FW-13 (Integration in Cognos Knowledge base)

ID	FW-14.
Name	Integration in Cognos Ontology.
Description	In the future, features and situations could be integrated with the Ontology, so that their identifiers are no longer strings but concepts with an associated semantical value.
Priority	Normal.

Table 7.14: Future work FW-14 (Integration in Cognos Ontology)

ID	FW-15.
Name	Full integration in Cognos.
Description	A full integration in the Cognos toolkit would enrich the annotation in other models by supplying circumstantial metadata to them.
Priority	Normal.

Table 7.15: Future work FW-15 (Full integration in Cognos)

ID	FW-16.
Name	Acquisition of knowledge of a real scenario.
Description	The Cognos toolkit may be used to acquire knowledge over real scenarios, so that this knowledge can feed a natural interaction system with a physical layer.
Priority	Low.

Table 7.16: Future work FW-16 (Acquisition of knowledge of a real scenario)

7.2.4 Natural Interaction System

Finally, some future works are proposed which have to do with the integration of the situation model in the natural interaction system, this is, in the cognitive architecture presented in figure 1.1.

ID	FW-17.
Name	Model installation.
Description	A new function will be included in the tool in order to install the simulation model. This function will install the database schema over an empty instance, so that it can be used to create and manage new networks.
Priority	High.

Table 7.17: Future work FW-17 (Model installation)

ID	FW-18.
Name	Integration with the Task Model.
Description	The situation model could enrich the Task Model by providing new available tasks for it to carry out.
Priority	High.

Table 7.18: Future work FW-18 (Integration with the Task Model)

ID	FW-19.
Name	Integration with the Dialogue Model.
Description	The situation model could be enriched by the Dialogue Model, as the last one could provide interactive strategies supporting the tasks of the first one.
Priority	High.

Table 7.19: Future work FW-19 (Integration with the Dialogue Model)

ID	FW-20.
Name	Full integration in the natural interaction system.
Description	The situation model could enhance the performance of any other model in the natural interaction system, as it can provide functionality to filter its knowledge using the circumstantial one.
Priority	Low.

Table 7.20: Future work FW-20 (Full integration in the natural interaction system)

ID	FW-21.
Name	Physical layer.
Description	So far, the simulation over the situation model provides emulator for a physical layer. Actually, this physical layer could be implemented to provide the system with the user location. Some interesting positioning technologies are GPS, RFID, GSM, UMTS, Wi-Fi, accelerometers or ultrasounds. Most of these methods work by triangulation, although other ones (such as accelerometers) estimate the changes on the location.
Priority	Low.

Table 7.21: Future work FW-21 (Physical layer)

ID	FW-22.
Name	Full evaluation of the situation model.
Description	A complete evaluation of the situation model could be performed after it has been integrated within the natural interaction system. This evaluation would test the system with and without the support of the situation model, and will observe the actual advantages that this model is providing to the interaction.
Priority	Low.

Table 7.22: Future work FW-22 (Full evaluation of the situation model)

This page has been intentionally left blank.

Appendix A

Scenarios

This appendix will illustrate some scenarios where the situation model can be used to assist the user, or to improve the user experience with other application.

A.1 Improving a Navigation System

This scenario shows how a pedestrian navigation system, which usually only considers the spatial aspect of the situation, can be enhanced where a broader view of the context is taken into account. For this case, we will consider that the system also considers the temporal and ambiental conditions of the material aspect of the situation.

Figure A.1 illustrates this scenario.

Martha (M) wants to go from the *Torres Quevedo* building to the *Sabatini* building, in the campus of Leganés of Universidad Carlos III. It is 9.00 PM and there is a heavy rain.

Usually, Martha would take the shortest route, which passes by the entry of the library and enters the *Sabatini* park with the fountain, just before going into the building by gate A. This route is shown in the figure as M-A.

However, there are some issues for which this route is not convenient. In the first place, entry A to *Sabatini* is closed after 5.00 PM, so Martha will not be able to access the building by this gate. Secondly, Martha did forget her umbrella and she really hates walking under the rain, something that wants to avoid as long as possible.

Martha, who is a very clever girl, starts her navigation system empowered by the situation model and asks for the best route. The system is perfectly aware of the context: it knows that entry A is closed after 5.00 PM, and it also knows that the weather is really annoying for Martha.

Finally, the system proposes an alternative route, M-B, which goes from Martha location

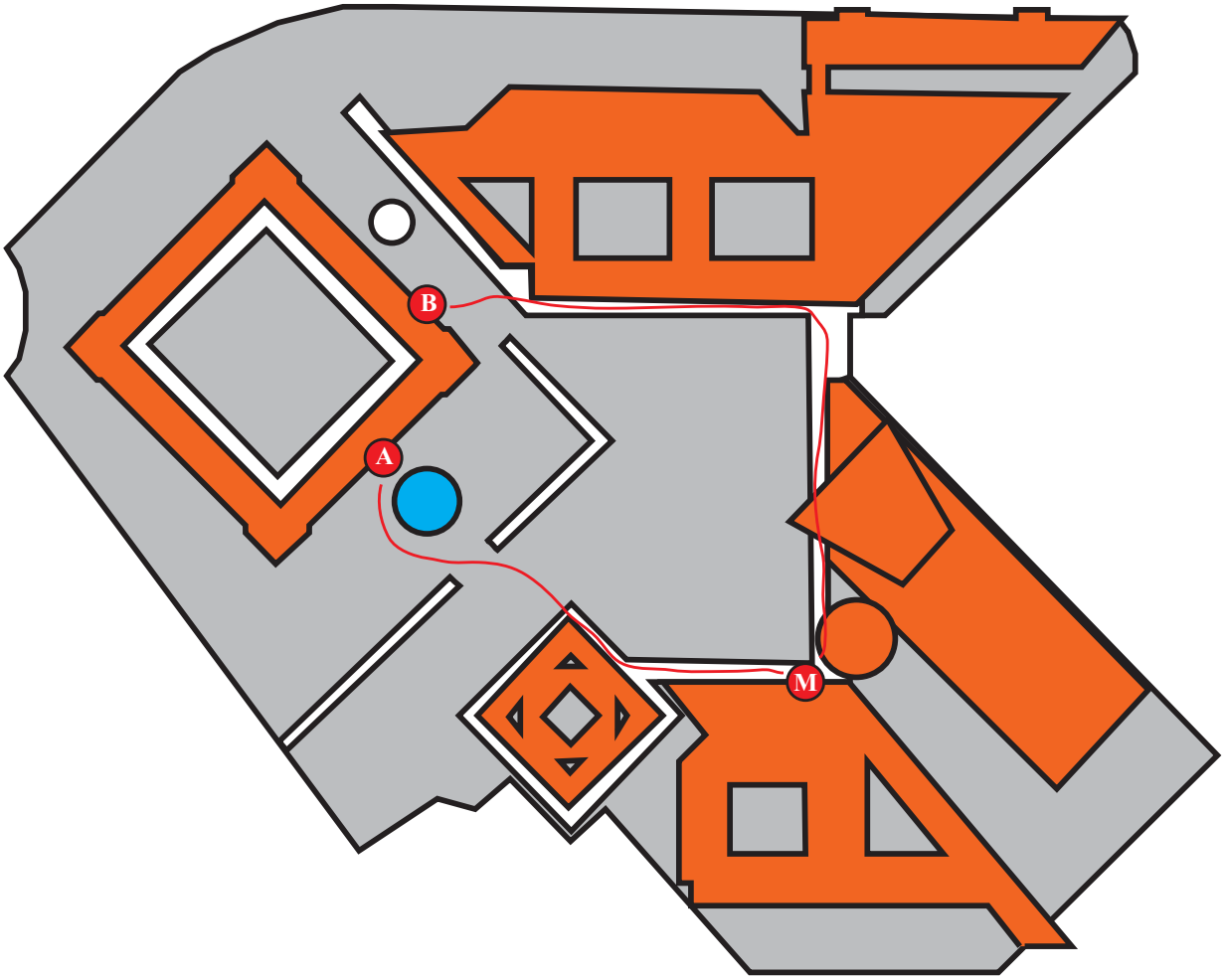


Figure A.1: Map of the campus illustrating the first scenario

to entry B, which still opens at 9.00 PM. Moreover, this route is covered by the buildings, so Martha will get to her goal dry and happy.

A.2 Enhancing a Natural Interaction System

This scenario shows a sample dialog between a user and a natural interaction system which is enriched by a situation model.

Arthur is in the building of *CafreSoft Corporation*, a young and promising software company. He has just finished a job interview and is really satisfied. It is 11.30 AM and it takes a few hours for him to go home, so he has decided to have his lunch in the building. He starts a dialogue with his personal assistant, Hal:

ARTHUR: Hi HAL! Look, I'd like to eat something here. What do you say?

[*The system realizes that it is 11.30 in the morning. It knows that* ARTHUR *usually have breakfast at 9.00 AM and lunch at 1.00 PM. The system asks* ARTHUR *for more information.*]

HAL: Nice! It's eleven thirty, would you rather have breakfast, or have lunch?

ARTHUR: Today I'll have something for lunch.

[*The system looks for places of interest near* ARTHUR.]

HAL: Ok. There is a snack bar in this floor, but it is only serving breakfast now. You can have a sandwich there. If you prefer, you can go to the restaurant to have a meal, and you can eat from the menu until 3 o'clock in the afternoon.

ARTHUR: Fine, let's go there.

[*The system tries to disambiguate the anaphora (there) by asking the user.*]

HAL: To the restaurant, right?

ARTHUR: Right.

HAL: Ok. Go ahead until the end of the corridor. Then you'll have to go to the first floor. You can take the elevator.

ARTHUR: Nice. By the way, do you know where can I find a bathroom?

[*The system looks for bathrooms. He finds a bathroom in the current floor, but it is only for the company staff. He looks for alternatives and assists* ARTHUR.]

HAL: There is a bathroom in this floor, but it is only for the personnel. However, you can use the restaurant restroom when you get there.

ARTHUR: Ok, I'll wait.

[ARTHUR *arrives to the end of the corridor and calls the elevator.*]

HAL: Remember, go to the first floor now.

ARTHUR: Thanks, HAL.

[ARTHUR *steps out of the elevator in the first floor.*]

HAL: Ok, turn right and you'll find the restaurant.

ARTHUR: Ok, thank's a lot!

A.3 Assisting Disabled People

This scenario shows how the situation model could take into account information about the user in order to improve its experience.

Alan is working on his University entrance exams these days. He has to do these exams in the campus of Leganés of Universidad Carlos III, and he does not know the college. He is a nice guy, but unfortunately he is using a wheelchair since he was five. Although he is really used to his wheelchair, he is still having some mobility issues.

Alan has downloaded in his smartphone a mobile application developed by the University wich contains the map of the campus and includes a simple navigation feature to guide the user to a searched location. Furthermore, from the last update a simple situation model was implemented which takes into account, among others, some information about the user. For instance, he could select in a profile that he is a disabled person, and the system will provide alternative routes.

In his case, the system will find a route that goes through ramps and elevators, rather than doing through stairs. This route will also prefer the automatic doors. Even if the route is longer and requires more time to get to the goal, the system will prefer it as it has less cost for Alan.

Appendix B

Ethical Considerations

In 1867, Alfred Nobel patented the dynamite, which was obtained as a result of adding *kieselguhr* to nitroglycerine, taking the form of a moldable paste. The invention increased significantly the safety and reduced the costs of mining works such as rock blasting, tunnels drilling, etc. However, dynamite also got an important role in the military industry. Nobel, which was very interested in social and peace issues and made a great fortune with his inventions, started the Nobel Foundation and, when he wrote his final will, he included a Prize for those persons or organizations which promoted science and peace. A summary of Nobel's life and work can be found in [NobelPrize.org, 2012].

Research in Artificial Intelligence (AI) often generates expectation, as it reveals new discoveries and inventions which show us realities which we were not able to imagine a few years ago. However, research in these areas is not controversy-free. On one side, advancements in AI systems helps to build a more modern and comfortable society. On the other, these intelligent systems are perceived as machines which will eventually replace humans at work. Regardless of whether these fears are unfounded or not, it is true that interest in AI is accompained by some distrust.

The purpose of this appendix is to evaluate the consequences of the development of this project over the society.

When scientists try to build computers that emulate the human behavior, we first have to model, in some way, our own knowledge; in order to introduce it into the computer. However, most of the times we are not aware of our own knowledge, as many things seems to be natural for us. In the last years, with the expansion of the social networks, we can realize that in some cases, the Internet actually stores more information about us that the one that we have. Even if all this information is spread in different computers, some *data mining* techniques can be applied to integrate it all and generate an impressively huge material about ourselves.

Although the situation model developed in this project is relatively simple, future works are intended to improve it in order to allow the support of several agents (for example, the

current user and also other people and objects) and the consideration of all the context aspects in the network graph, apart from the spatial dimensions.

In this case, the situation model could be storing a lot of confidential data about the users. Not only its current position, but also the evolution of its location, its role or its tasks over time. The model could then turn into the *Big Brother* that knows absolutely everything about everybody.

For this reason, it is important that the system does not store more information that the one that is absolutely required to provide the service. Furthermore, it is essential that the user is notified about the data that the system is storing about him and the reasons why this data is required, so that he can freely decide whether he accepts to use the system or not.

Appendix C

Evaluation Data

This chapter shows the original data gathered during the evaluation process, and it can be used to get a better idea on how the edition tool compares to the direct edition in terms of performance and accuracy, as well as what are the subjects' opinions regarding the two systems. From now on, *System A* is the name given to the system which allows direct edition of the model knowledge and *System B* is called to the edition tool developed in this project.

Table C.1 shows the execution time of both tasks for each of the experiment subjects.

	Direct Edition	**Edition Tool**
Subject 1	24' 29"	6' 14"
Subject 2	23' 34"	7' 28"
Subject 3	35' 46"	8' 34"
Subject 4	23' 36"	5' 3"
Subject 5	29' 20"	9' 22"
Subject 6	27' 12"	7' 26"

Table C.1: Time measurements for the evaluation

Tables C.2 and C.3 show the position of the nodes in the networks modeled by each of the subjects, for both the direct edition system and the edition tool respectively. This information has been used in the evaluation as a measure for the accuracy. The first row contains the original position for each of the nodes, while the others display the coordinates of the nodes in the networks modeled by each of the subject. Each cell is composed of two rows, containing the x and y coordinates of the node respectively.

	Direct Edition					
	Node 1	**Node 2**	**Node 3**	**Node 4**	**Node 5**	**Node 6**
	6.26	11.35	16.19	11.35	16.08	53.66
Original	6.39	11.39	11.39	15.93	49.53	15.93
	6.26	11.12	15.99	11.12	15.99	54.21
Subject 1	6.26	11.12	11.12	15.99	54.21	15.99
	6.58	11.84	15.79	11.84	15.79	53.29
Subject 2	6.67	11.34	11.18	16.00	50.03	15.34
	7.87	12.12	16.36	12.12	16.36	49.69
Subject 3	7.87	12.12	12.12	16.26	46.98	16.26
	6.90	11.73	14.75	9.99	14.65	54.51
Subject 4	6.66	9.99	11.73	16.56	48.62	15.70
	6.60	9.90	14.52	9.90	14.52	50.82
Subject 5	6.60	11.43	11.43	14.92	47.37	14.92
	6.09	11.06	15.76	11.13	15.76	53.31
Subject 6	5.34	9.77	9.79	13.85	43.68	13.85

Table C.2: Accuracy measurements for the evaluation with direct edition

	Edition Tool					
	Node 1	**Node 2**	**Node 3**	**Node 4**	**Node 5**	**Node 6**
	6.26	11.35	16.19	11.35	16.08	53.66
Original	6.39	11.39	11.39	15.93	49.53	15.93
	6.39	11.85	16.11	11.85	15.76	53.79
Subject 1	6.32	11.71	11.71	15.93	50.12	15.57
	6.28	10.78	15.88	10.90	15.76	53.67
Subject 2	6.41	11.07	11.19	16.08	49.30	15.50
	6.58	11.75	16.09	11.75	15.98	54.52
Subject 3	6.24	11.43	11.43	15.81	49.79	15.70
	6.47	12.12	16.24	12.12	16.24	54.94
Subject 4	6.57	11.62	11.62	16.18	50.35	16.38
	6.57	11.62	16.19	11.50	15.85	53.17
Subject 5	6.02	11.11	11.11	15.39	48.96	15.28
	6.38	11.58	16.31	11.70	16.19	53.78
Subject 6	6.39	11.28	11.39	15.69	49.53	15.81

Table C.3: Accuracy measurements for the evaluation with the edition tool

Finally, figures C.1 to C.6 contains the results of the surveys, which were filled by the subjects after completing the experiment. All the personal information has been removed from the survey in order to preserve the subjects confidentiality.

1. Evaluate the next statements in a scale from 1 to 5 (being 1 *strongly disagree* and 5 *strongly agree*) for each of the two systems.

Question	System A	System B
I felt good when performing the task (the system is comfortable).	1☐ 2☒ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☒
I found the functionality which I required easily (the system is intuitive).	1☒ 2☐ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☒
I think I completed the task successfully and without errors (the system is reliable).	1☐ 2☒ 3☐ 4☐ 5☐	1☐ 2☐ 3☒ 4☐ 5☐
I think the task did not take too much time to be completed (the system is agile).	1☒ 2☐ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☒

2. What do you think are the advantages of System A?

I think it's more fun, and also the fact of starting with a plane and created facilitates the beginning of the task

3. What do you think are the disadvantages of System A?

It is costly and imprecise. Also, it dosen't show graphically the results, making it more difficult to detect an error. Finally, the user must have some notion of creating the code.

4. What do you think are the advantages of System B?

It is more agile, more comfortable, faster and more precise. It shows graphically the results at all times. It is more intuitive because it is graphical.

5. What do you think are the disadvantages of System B?

It hasn't disadvantages.

6. In what situations do you think System A would be convenient?

Never

7. In what situations do you think System B would be convenient?

Always

8. Please, express here any other observation.

The interface is very nice.

Figure C.1: Survey completed by subject 1

1. Evaluate the next statements in a scale from 1 to 5 (being 1 *strongly disagree* and 5 *strongly agree*) for each of the two systems.

Question	System A	System B
I felt good when performing the task (the system is comfortable).	1☐ 2☐ 3☒ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☒
I found the functionality which I required easily (the system is intuitive).	1☐ 2☐ 3☒ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☒
I think I completed the task successfully and without errors (the system is reliable).	1☐ 2☐ 3☒ 4☐ 5☐	1☐ 2☐ 3☐ 4☒ 5☐
I think the task did not take too much time to be completed (the system is agile).	1☐ 2☐ 3☐ 4☐ 5☒	1☐ 2☐ 3☐ 4☐ 5☒

2. What do you think are the advantages of System A?

Repetition makes it a simple and mechanical system

3. What do you think are the disadvantages of System A?

All the data has to be manually introduced

4. What do you think are the advantages of System B?

It is intuitive, very graphical and easy to be used.

5. What do you think are the disadvantages of System B?

The screen is too large, It took some time to find some controls.

6. In what situations do you think System A would be convenient?

Don't know.

7. In what situations do you think System B would be convenient?

Don't know.

8. Please, express here any other observation.

Any.

Figure C.2: Survey completed by subject 2

1. Evaluate the next statements in a scale from 1 to 5 (being 1 *strongly disagree* and 5 *strongly agree*) for each of the two systems.

Question	System A	System B
I felt good when performing the task (the system is comfortable).	1☐ 2☒ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☒
I found the functionality which I required easily (the system is intuitive).	1☒ 2☐ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☒ 5☐
I think I completed the task successfully and without errors (the system is reliable).	1☒ 2☐ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☒ 5☐
I think the task did not take too much time to be completed (the system is agile).	1☒ 2☐ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☒

2. What do you think are the advantages of System A?

3. What do you think are the disadvantages of System A?

It's very easy to make mistakes setting the coordinates. It's needed a more precisely rule (with more than one decimal position). You have to spend a lot of time taking the measures because usually you get wrong and you have to repeat the measure. You have to write the pl/sql code and in this moment you can get wrong again transcribing the measures.

4. What do you think are the advantages of System B?

It's easy to create the network in a few minutes with less errors than the manual system. You don't have to take the measures manually and only moving the mouse. You can modify the network easily.

5. What do you think are the disadvantages of System B?

You cannnot set a node in a known coordenates

6. In what situations do you think System A would be convenient?

When you don't have access to an informatic system, when you hate computers...

7. In what situations do you think System B would be convenient?

I that is convenient allways you have an informatic system

8. Please, express here any other observation.

Figure C.3: Survey completed by subject 3

1. Evaluate the next statements in a scale from 1 to 5 (being 1 *strongly disagree* and 5 *strongly agree*) for each of the two systems.

Question	System A	System B
I felt good when performing the task (the system is comfortable).	1☐ 2☐ 3☐ 4☒ 5☐	1☐ 2☐ 3☐ 4☐ 5☒
I found the functionality which I required easily (the system is intuitive).	1☐ 2☐ 3☐ 4☒ 5☐	1☐ 2☐ 3☐ 4☒ 5☐
I think I completed the task successfully and without errors (the system is reliable).	1☐ 2☐ 3☐ 4☐ 5☒	1☐ 2☐ 3☐ 4☐ 5☒
I think the task did not take too much time to be completed (the system is agile).	1☐ 2☐ 3☒ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☒

2. What do you think are the advantages of System A?

It provides an API an thus can be used by other systems or scripts.

3. What do you think are the disadvantages of System A?

Too much typing, users have to look up the API and must have basic programming skills.

4. What do you think are the advantages of System B?

GUIs are easier to understand.

5. What do you think are the disadvantages of System B?

No scripting capabilities.

6. In what situations do you think System A would be convenient?

Whenever there is a repetitive and scriptable task.

7. In what situations do you think System B would be convenient?

For unexperienced users or for defining a network quickly.

8. Please, express here any other observation.

Figure C.4: Survey completed by subject 4

1. Evaluate the next statements in a scale from 1 to 5 (being 1 *strongly disagree* and 5 *strongly agree*) for each of the two systems.

Question	System A	System B
I felt good when performing the task (the system is comfortable).	1☒ 2☐ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☒ 5☐
I found the functionality which I required easily (the system is intuitive).	1☐ 2☒ 3☐ 4☐ 5☐	1☐ 2☐ 3☒ 4☐ 5☐
I think I completed the task successfully and without errors (the system is reliable).	1☐ 2☐ 3☒ 4☐ 5☐	1☐ 2☐ 3☒ 4☐ 5☐
I think the task did not take too much time to be completed (the system is agile).	1☐ 2☒ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☒ 5☐

2. What do you think are the advantages of System A?

I do not find advantanges in this system

3. What do you think are the disadvantages of System A?

I think as more important the inprecision and lack of scalability.

4. What do you think are the advantages of System B?

More intuitive, usable and faster.

5. What do you think are the disadvantages of System B?

Some elements are confuse. The reference points (triangle) and circles are treated the similar way to move. The zoom is different for points and the image or plane. I think the zoom should be integrated for both.

6. In what situations do you think System A would be convenient?

I dont know

7. In what situations do you think System B would be convenient?

I think the system B is convenient for the purpose that the tooal has been created.

8. Please, express here any other observation.

I suggest changing the zoom.

Figure C.5: Survey completed by subject 5

1. Evaluate the next statements in a scale from 1 to 5 (being 1 *strongly disagree* and 5 *strongly agree*) for each of the two systems.

Question	System A	System B
I felt good when performing the task (the system is comfortable).	1☐ 2☒ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☒
I found the functionality which I required easily (the system is intuitive).	1☐ 2☐ 3☐ 4☐ 5☒	1☐ 2☐ 3☐ 4☐ 5☒
I think I completed the task successfully and without errors (the system is reliable).	1☐ 2☐ 3☒ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☒
I think the task did not take too much time to be completed (the system is agile).	1☒ 2☐ 3☐ 4☐ 5☐	1☐ 2☐ 3☐ 4☐ 5☒

2. What do you think are the advantages of System A?

Low level edition (close to tech) often enables greater datail and process shortcuts. Copy paste is also powerful, and I can keep the generated code (and run it as many times as desired).

3. What do you think are the disadvantages of System A?

Easy mismatch, hard doing,

4. What do you think are the advantages of System B?

Automate, powerful; enables edition of larger scenarios, enables minor granularities (greater level of detail).

5. What do you think are the disadvantages of System B?

Copy paste of nodes; alignment of nodes (easy by manual edition); vertical/horizontal distribution (easy by manual edition)

6. In what situations do you think System A would be convenient?

small scale problems, with regular grids

7. In what situations do you think System B would be convenient?

any problem of this kind

8. Please, express here any other observation.

please, support the author! (and also his tutor). Bank notes are accepted (preferable the purple ones).

Figure C.6: Survey completed by subject 6

This page has been intentionally left blank.

Appendix D

Installation Guide

The management and simulation tool is provided as a `.jar`, a *Java archive*, which can be directly executed in a system with the Java Virtual Machine (JVM) installed. Moreover, as the database is located in a server within the network of Universidad Carlos III, a connection to the university Virtual Private Network (VPN) must be established as well.

This chapter explains the steps in order to install the JVM, establish the VPN connection and start the application.

D.1 Installing the JVM

The Java environment can be downloaded from the next site: `http://www.java.com`. Follow the instructions in order to download the installation files. The website will detect automatically your operating system and will provide the last version of the JVM.

Once the installation package is downloaded, load it and follow the installation instructions.

D.2 Connecting to the VPN

Although the VPN connection is not required to load the application, it will be required in order to connect to the default database if the application is running outside the university network.

The next page contains a manual for establishing the VPN connection from different operating systems: `https://asyc.uc3m.es/index.php?Id=168`.

D.3 Loading the Application

After the Java installation process concludes, you will be able to open the `.jar` file containing the management and simulation tool. In some systems, double-clicking on the file must be enough to open the application. However, other operating systems may require to open a command window, place the bash directory where the `.jar` file is contained and execute the next command: `java -jar cognos_s.jar`.

Appendix E

User Manual

This section will explain the use of the tool developed for this project. The design and structure for this application was discussed in chapter 4.

As that chapter explained, the application contains four different screens, each of them in a different tab. This manual describes the functionality provided in each of the screens.

E.1 Main Screen

Figure E.1 shows an screenshot of the main screen, which provides functionality for establishing and closing a connection to a database.

The components of the figure are explained here:

1. The **tab selector** allows to move between the different screens of the application. The tabs will remain disabled unless a connection to a database is established.
2. The **database connection** panel contains all the required fields to establish a connection to an Oracle database. If the input in any of the fields is wrong, its background color will change to light red to indicate this fact.
3. The **connect** button establishes the connection to the database.
4. The **disconnect** button closes the connection to the database that was already established.
5. The **status bar** shows the status of the connection, or an error message which will be painted in red if any occurs.

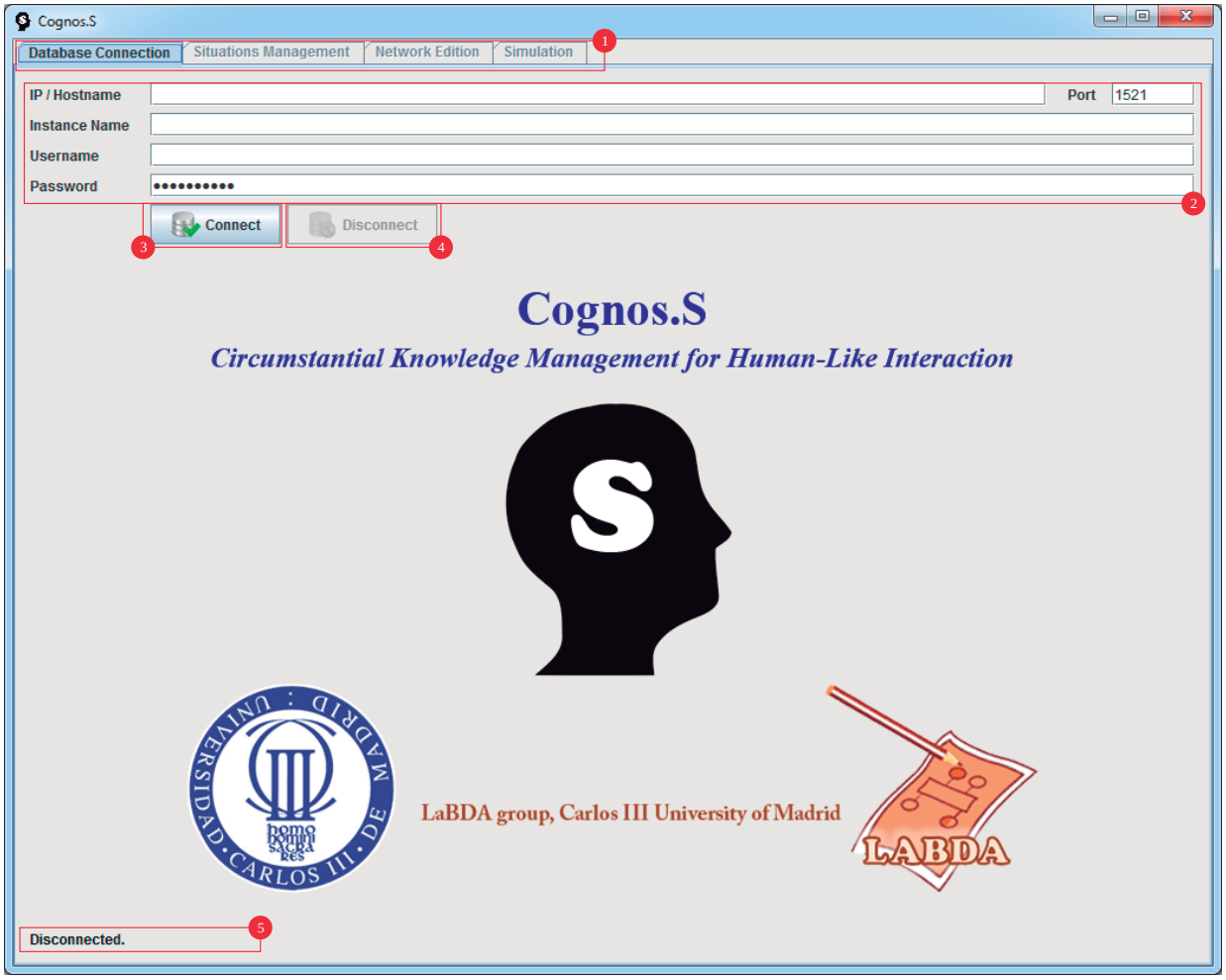


Figure E.1: Screenshot for the main screen

E.2 Situations Management

This screen allows the management of situations and features. Figure E.2 shows an screenshot of this interface.

The components are described here:

1. The **situational aspect** selector allows to choose one situational aspect.
2. The **situations list** enumerates all the situations for the selected situational aspect in (1). When a situation is selected, its description is shown just under the list.
3. The **situation removal button** deletes the situation selected in (2) from the system.
4. The **new situation panel** allows to add a new situation with the specified name and description. In order to make the insertion effective, the button must be pressed.

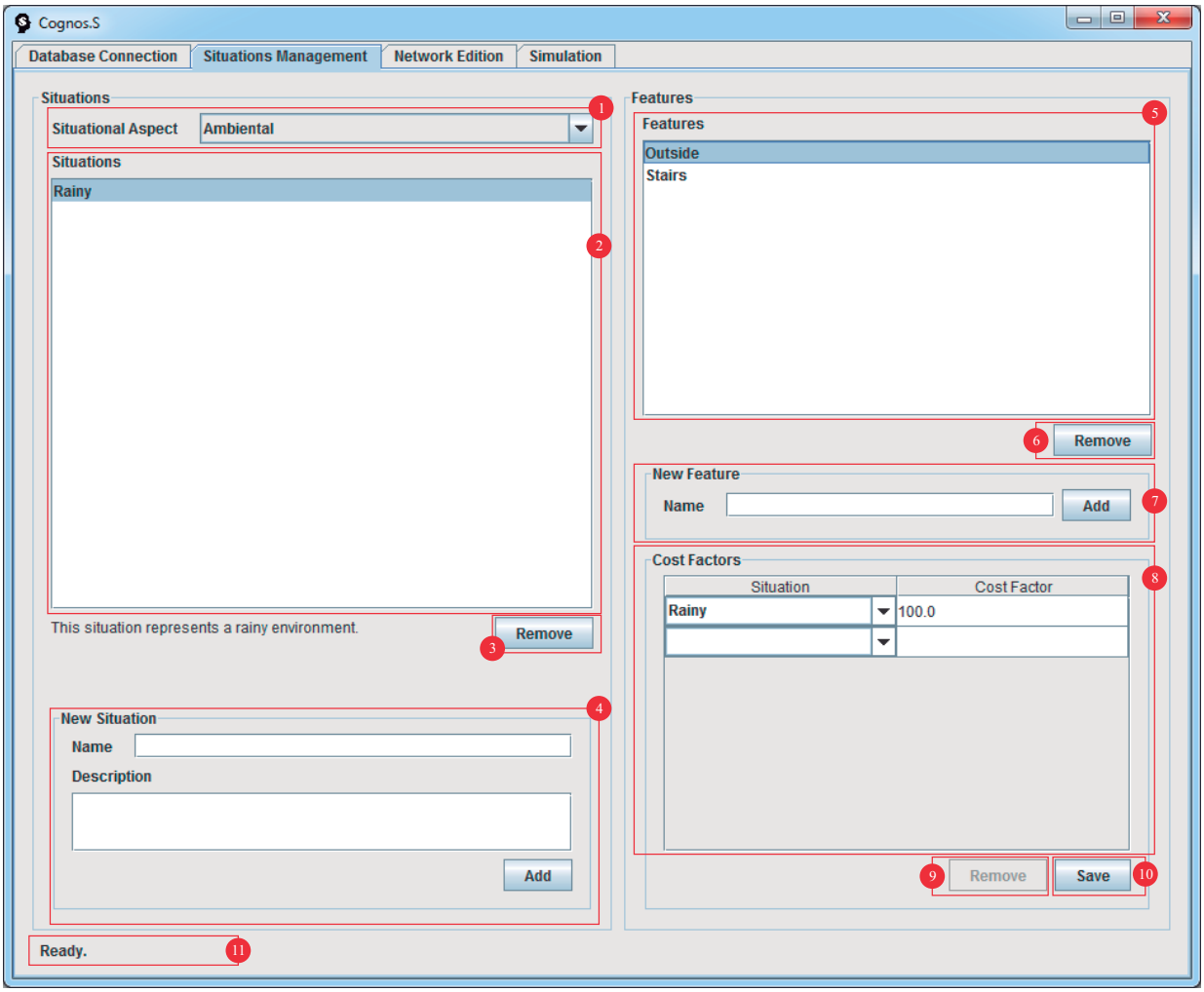


Figure E.2: Screenshot for the situations management screen

The new situation will be added in the situational aspect selected in (1), and shown in the list (2).

5. The **features list** shows all the features stored in the system.

6. The **remove feature button** deletes the feature selected in (2) from the system.

7. The **add feature panel** allows the insertion of a new feature, with the specified name, in the system. The new feature will be immediately shown in the list (5).

8. The **cost factors** table shows the cost factors for the feature selected in (5). The first column contains the situation, while the second specifies the cost as a positive real number. When a new cost factor is added, a new empty row is included in the table. The selector in the first column lists all the situations in the system. Notice that the system will not allow two different cost factors referring to the same situation.

9. The **cost factor removal button** deletes the row with the selected cost factor from table (8).

10. The **cost factor save button** confirms the changes in the cost factors shown (8) and saves them in the database.

11. The **status bar** shows the status of the last performed operation, or whether an error occurred.

E.3 Network Management

This screen allows to manage the networks of the situation model. Figure E.3 shows this interface.

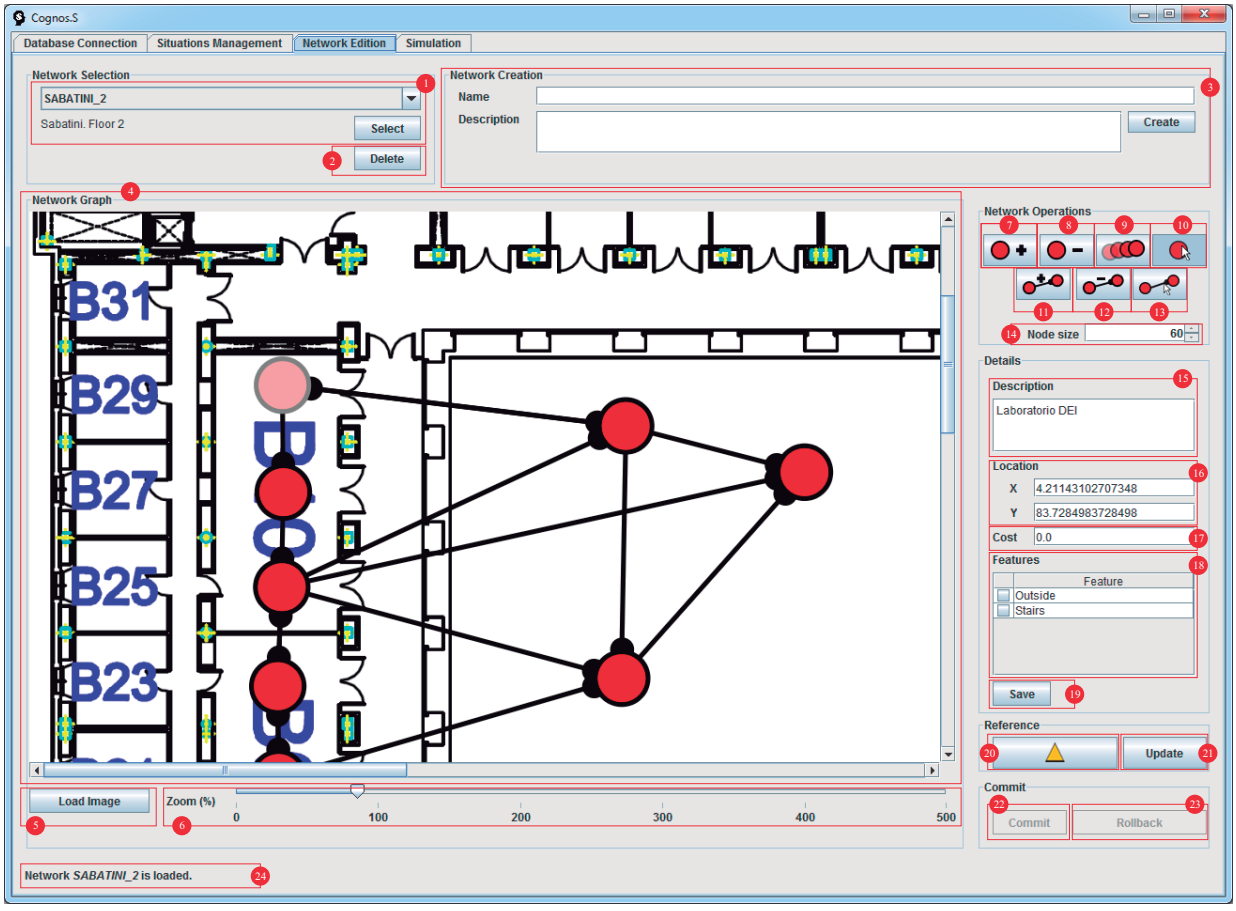


Figure E.3: Screenshot for the network management screen

The components of this screen are detailed here:

1. The **network selection** panel allows to choose an existing network. When a network is selected in the box, its description is shown. The button *select* must be pressed in order to load the network.

2. The **network removal button** deletes the network selected in the box (1) from the system.

3. The **network creation panel** allows to create a new network with the specified name and description. The network is created when the *create* button is pressed.

4. The **map area** shows the network plane and the network items (nodes and links). The picked elements are highlighted.

5. The **load image button** shows a dialog to choose a background image for the network plane.

6. The **zooming slider** allows to zoom in and zoom out the map area (4). The number under the slider is the zoom percentage over the image real size.

7. The **node creation button** will create a node in the place where the user clicks over the map area (4).

8. The **node removal button** allows the user to select a node in the map area (4) to be deleted. Moreover, its input and output links will be removed as well.

9. The **node motion button** allows the user to move a node from one position of the plane to another. The user must drag and drop the node in the map area (4).

10. The **node selection button** allows the user to select a node from the map area (4). When the user selects a node, its details are shown in the details panel (15), (16), (17) and (18).

11. The **link creation button** will create a link between two nodes. The user must click first in the source node and later in the target node, in the map area (4) in order to create the link.

12. The **link removal button** will delete a link between two nodes. In order to select the link to be removed, the user must first click in the source node and later in the target node, in the map area (4).

13. The **link selection button** allows the user to select a link from the map area (4), by clicking first in the source node and later in the target node. When the link is selected, its details are shown in the details panel (15), (17) and (18). The location details (16) are not shown for links.

14. The **description details** are shown when either a node or a link is selected, and contains the item description.

15. The **location details** are shown when a node is selected, and contains the coordinates of the node position.

16. The **cost details** are shown when either a node or a link is selected, and contains the basic cost for the network item.

17. The **features details** are shown when either a node or a link is selected. This element contains a table with features, each of them with a checkbox that indicates whether the network item has that feature.

18. The **save details button** saves the changes that the user does over the details fields (15), (16), (17) and (18).

19. The **reference node creation button** allows to create a new reference node, by clicking over the map area (4) after selecting the button. This reference node is considered as an usual node (and the same operations can be applied over them), but is represented as a yellow triangle in the amp area (4). At most two reference nodes can be created.

20. The **update references button** will update the coordinates of all the nodes in the network automatically. For this to happen, the plane must contain two reference nodes whose coordinates must be set.

21. The **commit button** will persistently store all the changes done since the last *commit* or *rollback* operation in the database. It must be noticed that if there are some uncommited changes and the user tries to perform any operation that would discard these changes (such as disconnecting from the database, loading another network or even closing the application) a warning will appear, such as the one shown in figure E.4.

22. The **rollback button** will discard all the changes done since the last *commit* operation.

23. The **status bar** will show the status of the executed operations, as well as any error which could have occurred (in red).

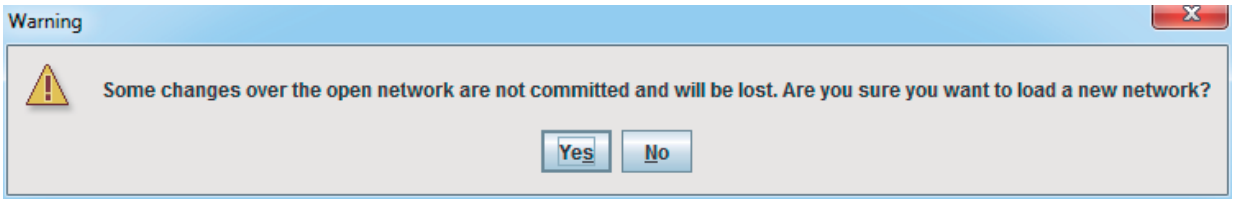


Figure E.4: Sample warning message for uncommited changes

E.4 Simulation

This screen allows to carry out a simulation over a network from the situation model. Figure E.5 shows this part of the application.

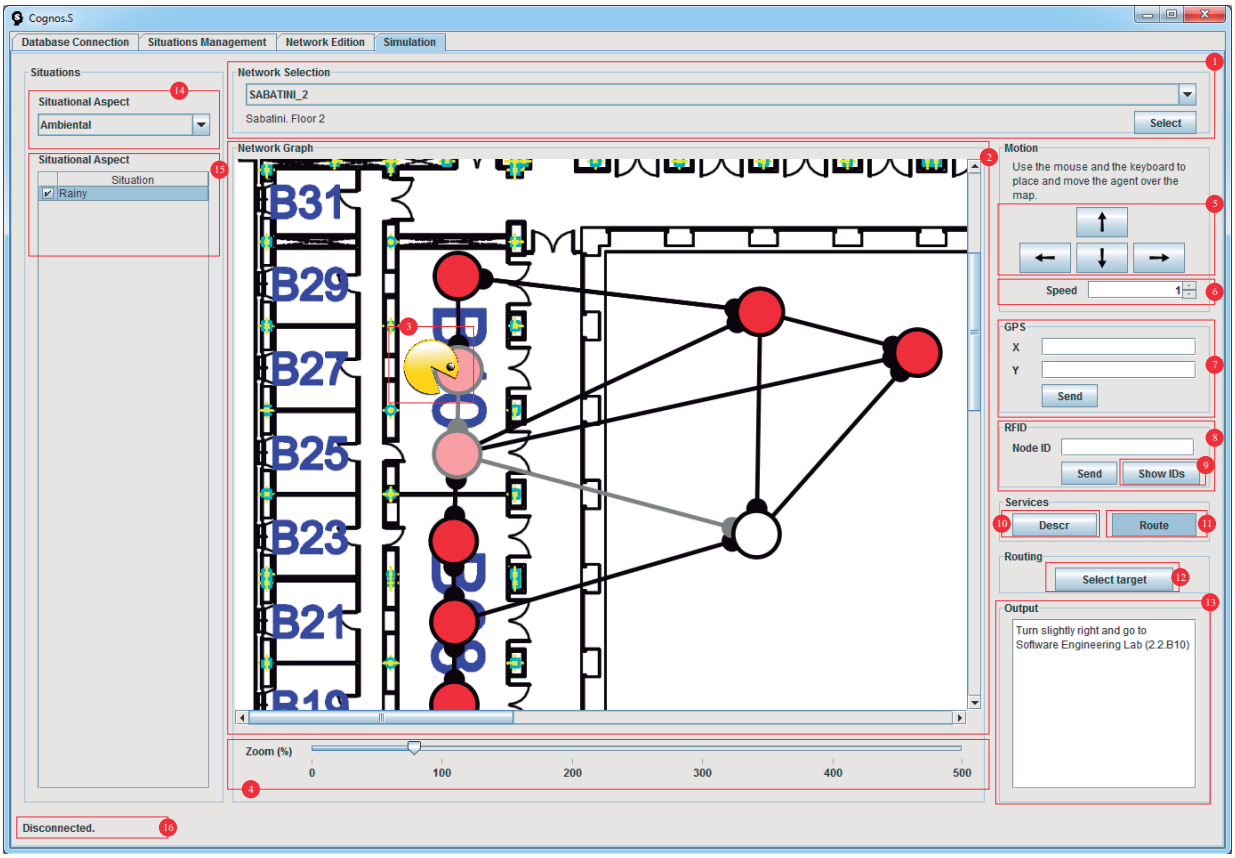


Figure E.5: Screenshot for the simulation screen

The components of this screen are described here:

1. The **network selection panel** allows to select a network from the selector box. When a network is chosen, its description is shown under the box. The button must be clicked in order to load the network for the simulation.

2. The **map area** shows the plane for the network, with all its items (nodes and links). The items which are picked (such as the closest node to the agent or the nodes and links of a route) are highlighted. Moreover, the target node (if any) is shown in white. Clicking on this area will move the user node (3) to the clicked position. Also, the direction keys of the keyboard can be used in order to move the user location.

3. The **user node** shows the position of the user over the map. This node is rotated according to the user orientation (eight different values are possible).

4. The **zooming slider** allows to zoom in and zoom out the map area (2). The number under the slider is the zoom percentage over the image real size.

5. The **motion keys** are an alternative for the user (3) motion, which substitutes the direction keys in the keyboard.

6. The **speed spinner** allows to increase or decrease the motion speed of the user (3) over the map area (2).

7. The **GPS emulator panel** allows to input two coordinates (X and Y) and, by clicking int the *send* button, send the user (3) to the location in the map (2) corresponding to those coordinates

8. The **RFID emulator panel** allows to input a node ID and, by clicking in the *send* button, move the user to the corresponding node.

9. The **RFID show IDs button** shows the ID over each node in the map area (3). This button can be pressed again to hide those IDs.

10. The **description service button** executes the description service as long as the button is pressed.

11. The **routing service button** executes the routing service as long as the button is pressed.

12. The **select target button** must be used along with the routing service (11) to select a target node in the map area (2).

13. The **service output textbox** shows the output of the service in execution.

14. The **situational aspect selector** allows to choose a situational aspect. When one is selected, the situations within that aspect are shown in the situations chooser (16).

15. The **situations chooser** shows the situations corresponding to the situational aspect selected in (15). A checkbox is available for each situation, so that checking or unchecking it determines whether that situation applies or not for the simulation.

16. The **status bar** shows the status of the last operation performed, or a message in red if an error occurred.

Glossary

AI-hard the set of the most difficult AI problems, whose complexity is equivalent to making computer as intelligent as people. 26

ambient intelligence term to refer to the idea that a computer system is incorporated to an environment in order to sense it and act proactively to provide some benefits to the users. 21, 22, 25

circumstance piece of contextual information. 13

circumstantial knowledge knowledge regarding the context. 13

context set of circumstances applying in an environment, which surround and condition an event. 23

context model computational model that aims to formalize and store the context of a specific system in order to ease its processing. 23, 31

context-aware system system that takes into consideration contextual information in order to adapt its behavior to the current context. 21–23, 25

Dialogue Model model of the natural interaction system that processes the interlocutor communicative acts and develops its own acts and conversational strategies in order to generate a discourse. 10, 15, 113

human-like interaction style of Human-Computer Interaction which tries to resemble the interaction between humans. 13–15, 33, 138

implicit interaction set of actions that are not primarily aimed to be part of an interaction, but which another system understands in order to generate a response. 25

knowledge base database which serves for knowledge management. 69, 70, 72, 73, 92, 93, 98

knowledge model model that stores some kind of formalized knowledge expressed in some knowledge representation language, so that it can be interpreted by a computer. 69

location-aware system context-aware system only dealing with the spatial aspect of the context. 23

mockup high-level design of user interfaces, which shows the user what the software will look like before starting its implementation. 72, 73, 75

multimodality fact of using several modalities for the communication, such as speech, gestures, written language, etc. 26, 33

natural interaction *see human-like interaction.* 10, 25, 27, 28, 33, 35, 36, 105–107, 112, 113, 137, 138

Natural Language Processing field of AI which groups a set of computational techniques for dealing with natural language. 25, 26, 139

ontology system that formally represents the knowledge of a domain as a set of concepts and the relationships among them, supporting the other models of the natural interaction architecture. 10, 15, 111

pervasive computing term to refer to the fact that computers are located everywhere, so the user could connect to the Internet at any time. 20, 21

situation model computational model that aims to formalize and store the circumstantial knowledge regarding any interaction in order to process it. 10, 14, 19, 28, 36, 69, 106, 107, 111–113

spatio-temporal database database which manages space and time information. 15

ubiquitous computing term coined by Mark Weiser in the late 1980s to refer to a computing paradigm in which computers vanish in the background, and the user can focus on the purpose of the interaction, rather than in its form. 19–22, 25, 33

User Model model of the natural interaction system that formalizes the knowledge about the user preferences, in order to provide a customized interaction to the user. 15

Acronyms

ACO Ant Colony Optimization. 87, 110

AI Artificial Intelligence. 21, 22, 25, 26, 28, 119, 137, 138

CLI Command Line Interface. 25–27

CRUD Create, Read, Update and Delete. 68, 78

ER Entity-Relationship. 5, 24, 69–71

GPS Global Positioning System. 23, 44, 55, 76, 113

GUI Graphical User Interface. 25–27, 30, 68, 72, 83, 89, 90

HCI Human-Computer Interaction. 20, 25

IDE Integrated Development Environment. 78

JUNG Java Universal Network/Graph Framework. 91

LaBDA Advanced Databases Group. 16, 78, 106

NLP Natural Language Processing. 25, 26

NUI Natural User Interface. 26, 27, 33

OCR Optical Character Recognition. 26

PC Personal Computer. 20

PDA Personal Digital Assistant. 20

RFID Radio Frequency IDentification. 44, 55, 76, 113

UML Unified Modeling Language. 24

VPN Virtual Private Network. 127

WYSIWYG What You See Is What You Get. 78

Bibliography

[Allen et al., 1994] Allen, J. F., Schubert, L. K., Ferguson, G., Heeman, P., hee Hwang, C., Kato, T., Light, M., Martin, N. G., Miller, B. W., Poesio, M., and Traum, D. R. (1994). The TRAINS Project: A Case Study in Building a Conversational Planning Agent. *Journal of Experimental and Theoretical AI*, 7:7–48.

[Apple.com, 2012] Apple.com (2012). Using Siri. Do I have to say things a certain way to get Siri to respond? `http://www.apple.com/iphone/features/siri-faq.html`. Accessed March 6th, 2012.

[Baldauf et al., 2007] Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A Survey on Context-Aware Systems. *International Journal of Ad Hoc and Ubiquitous Computing*, pages 263–277.

[Bernsen, 2000] Bernsen, N. O. (2000). What is Natural Interactivity? In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, pages 34–37.

[Bill N. Schilit and Want, 1994] Bill N. Schilit, N. A. and Want, R. (1994). Context-Aware Computing Applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85–90.

[Blocher, 2006] Blocher, A. (2006). Facts and Figures About the SmartKom Project. In *SmartKom: Foundations of Multimodal Dialogue Systems*, chapter 2, pages 29–39. Springer.

[BOE, 2008] BOE (January 19th, 2008). Real Decreto 1720/2007, de 21 de diciembre, por el que se aprueba el Reglamento de desarrollo de la Ley Orgánica 15/1999, de 12 de diciembre, de protección de datos de carácter personal. Boletín Oficial del Estado, 17:4103-4136.

[Boehm, 1986] Boehm, B. (1986). A Spiral Model of Software Development and Enhancement. *ACM SIGSOFT Software Engineering Notes*, 11:14–24.

[Bolt, 1980] Bolt, R. A. (1980). Put-That-There: Voice and Gesture at the Graphics Interface. *ACM Computer Graphics*, 14:262–270.

[Calle, 2004] Calle, F. J. (2004). *Interacción Natural mediante Procesamiento Intencional: Modelo de Hilos en Diálogos*. PhD thesis, Universidad Politécnica de Madrid.

[Calle et al., 2010] Calle, F. J., Albacete, E., Sánchez, E., del Valle, D., Rivero, J., and Cuadra, D. (2010). Cognos: A Natural Interaction Knowledge Management Toolkit. In *Natural Language Processing and Information Systems*, pages 303–304. Springer.

[Calle et al., 2011] Calle, F. J., Albacete, E., Sánchez, E., Olaziregi, G., del Valle, D., Rivero, J., and Cuadra, D. (2011). Cognos: A Pragmatic Annotation Toolkit for the Acquisition of Natural Interaction Knowledge. *Procesamiento del Lenguaje Natural*, pages 269–276.

[Calle et al., 2008] Calle, F. J., Martínez, P., del Valle, D., and Cuadra, D. (2008). Towards the Achievement of Natural Interaction. In *Engineering the User Interface. From Research to Practice*, pages 63–80. Springer.

[Chomsky, 1957] Chomsky, N. (1957). *Syntactic Structures*. Mouton & Co.

[Clements et al., 2003] Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., and Stafford, J. (2003). *Documenting Software Architectures: Views and Beyond*. Addison-Wesley.

[Connolly, 2001] Connolly, J. (2001). Context in the Study of Human Languages and Computer Programming Languages: A Comparison. In *Modeling and Using Context*, volume 2116, pages 116–128. Springer.

[Cook et al., 2007] Cook, D. J., Augusto, J. C., and Jakkula, V. R. (2007). Ambient Intelligence: Technologies, Applications, and Opportunities.

[Cuadra et al., 2008] Cuadra, D., Calle, F. J., Rivero, J., and del Valle, D. (2008). Applying Spatio-Temporal Databases to Interaction Agents. In *International Symposium on Distributed Computing and Artificial Intelligence*, pages 536–540. Springer.

[Dey and Abowd, 1999] Dey, A. K. and Abowd, G. D. (1999). Towards a Better Understanding of Context and Context-Awareness. In *Proceedings of the 1st international Symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271.

[Ducatel et al., 2001] Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., and Burgelman, J. C. (2001). Scenarios for Ambient Intelligence in 2010. Technical report, IST Advisory Group.

[Eckerson, 1995] Eckerson, W. W. (1995). Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications. *Open Information Systems*, 10(1).

[Eisaku and Yasuhiru, 2006] Eisaku, M. and Yasuhiru, M. (2006). Steps Towards Ambient Intelligence. *NIT Technical Review*, pages 50–55.

[Ferguson and Allen, 1998] Ferguson, G. and Allen, J. F. (1998). TRIPS: An Integrated Intelligent Problem-Solving Assistant. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 567–572. AAAI Press.

[Gee, 1999] Gee, J. P. (1999). *Introduction to Discourse Analysis*. Routledge.

[Hart et al., 1968] Hart, P., Nilsson, N., and Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

[Henricksen and Indulska, 2005] Henricksen, K. and Indulska, J. (2005). Developing Context-Aware Pervasive Computing Applications: Models and Approach. *Pervasive and Mobile Computing*, pages 37–64.

[Herzog and Reithinger, 2006] Herzog, G. and Reithinger, N. (2006). The SmartKom Architecture: A Framework for Multimodal Dialogue Systems. In *SmartKom: Foundations of Multimodal Dialogue Systems*, chapter 4, pages 55–70. Springer.

[IDC, 1996] IDC (1995-1996). Transition to the Information Highway Era. *Information Industry and Technology Update*, page 2.

[IEEE, 1998] IEEE (1998). 830-1998 - IEEE Recommended Practice for Software Requirements Specifications. Institute for Electrical and Electronics Engineers.

[Izquierdo, 2011] Izquierdo, P. J. (2011). Entorno Multiagente Implementado sobre Oracle 11g. Bachelor thesis, Universidad Carlos III de Madrid.

[James Allen and Stent, 2001] James Allen, g. F. and Stent, A. (2001). An Architecture for More Realistic Conversational Systems. In *Proceedings of Intelligent User Interfaces 2001*, pages 17–17.

[Jones, 1999] Jones, S. (1999). Digital Era will Make Prime-Time Television Obsolete. *Financial Times*, page 10.

[Korf, 1985] Korf, R. E. (1985). Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, 27:97–109.

[Liddy, 2003] Liddy, E. D. (2003). Natural Language Processing. *Encyclopedia of Library and Information Services*.

[Lohr and Markoff, 1998] Lohr, S. and Markoff, J. (1998). Computing's Next Wave is Nearly at Hand; Imagining the Future in a Post-PC World. *The New York Times*, page C1.

[Malaka and Zipf, 2000] Malaka, R. and Zipf, A. (2000). Challenging IT Research in the Framework of a Tourist Information System. In *Information and Communication Technologies in Tourism 2000*, pages 15–27. Springer.

[McCarthy and Buvac, 1997] McCarthy, J. and Buvac, S. (1997). Formalizing Context (Expanded Notes). In *Working Papers of the AAAI Fall Symposium on Context in Knowledge Representation and Natural Language*, pages 99–136. American Association for Artificial Intelligence.

[NobelPrize.org, 2012] NobelPrize.org (2012). Alfred Nobel - His Life and Work. `http://www.nobelprize.org/alfred_nobel/biographical/articles/life-work/`. Accessed February 16th, 2012.

[Oracle, 2012] Oracle (2012). Oracle Spatial Java API Reference 10g. Documentation for class JGeometry. `http://docs.oracle.com/cd/B19306_01/appdev.102/b14373/oracle/spatial/geometry/JGeometry.html`. Accessed May 30th, 2012.

[Oviatt and Cohen, 2000] Oviatt, S. and Cohen, P. (2000). Multimodal Interfaces that Process what Comes Naturally. *Communications of the ACM*, 43:45–53.

[Porzel et al., 2006] Porzel, R., Gurevych, I., and Malaka, R. (2006). In Context: Integrating Domain- and Situation-Specific Knowledge. In *SmartKom: Foundations of Multimodal Dialogue Systems*, chapter 18, pages 269–284. Springer.

[Potts, 1987] Potts, M. (1987). Computer Industry Wary of Jobs-Perot Alliance. *The Washington Post*, page H2.

[Rivero, 2012] Rivero, J. (2012). *Búsqueda Rápida de Caminos en Grafos de Alta Cardinalidad Estáticos y Dinámicos*. PhD thesis, Universidad Carlos III de Madrid.

[Rivero et al., 2007] Rivero, J., Cuadra, D., del Valle, D., and Calle, F. J. (2007). Incorporating Circumstantial Knowledge Influence over Natural Interaction. In *IEEE International Conference on Pervasive Services*, pages 415–420.

[Ronzani, 2009] Ronzani, D. (2009). The Battle of Concepts: Ubiquitous Computing, Pervasive Computing and Ambient Intelligence in Mass Media. *Ubiquitous Computing and Communication Journal*, 4:9–19.

[Schmidt, 2000] Schmidt, A. (2000). Implicit Human Computer Interaction Through Context. *Personal and Ubiquitous Computing*, 4:191–199.

[Schofield, 1994] Schofield, J. (1994). Analysis: Can Novell Make the Connections? *The Guardian (London)*, page T8.

[Smith, 1985] Smith, H. C. (1985). Database Design: Composing Fully Normalized Tables from a Rigorous Dependency Diagram. *Communications of the ACM*, 28(8).

[Sánchez-Pi, 2011] Sánchez-Pi, N. (2011). *Intelligent Techniques for Context-Aware Systems Adaptation*. PhD thesis, Universidad Carlos III de Madrid.

[Strang and Linnhoff-Popien, 2004] Strang, T. and Linnhoff-Popien, C. (2004). A Context Modeling Survey. In *Workshop on Advanced Context Modelling, Reasoning and Management*, pages 1–8.

[Turing, 1950] Turing, A. M. (1950). Computer Machinery and Intelligence. *Mind*, 59:433–460.

[Wahlster, 2006] Wahlster, W. (2006). Dialogue Systems Go Multimodal: The SmartKom Experience. In *SmartKom: Foundations of Multimodal Dialogue Systems*, chapter 1, pages 3–27. Springer.

[Want et al., 1992] Want, R., Hopper, A., Falcao, V., and Gibbons, J. (1992). The Active Badge Location System. *ACM Transactions on Information Systems*, 10:91–102.

[Weiser, 1991] Weiser, M. (1991). The Computer for the 21st Century. *Scientific American*, 265:94–104.

[Weiser and Brown, 1996] Weiser, M. and Brown, J. S. (1996). The Coming Age of Calm Technology.

This page has been intentionally left blank.