Universidade Nova de Lisboa

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Electrotécnica

Design and implementation of an autonomous, proactive, and reactive software infrastructure to help improving the management level of projects.

Por

João Pedro Dias Antunes

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para obtenção do grau de Mestre em Engenharia Electrotécnica e de Computadores

Orientador: Professor Celson Lima

Lisboa

2010

Dedico esta dissertação à minha família e amigos.

Acknowledgement

I want to express my thanks to Professor Celson Lima for the idea of this work, and for all the support he always provided me during the preparation of this dissertation. I also extend my thanks to those who supported me and encouraged me during this process, particularly to my parents João Carlos Antunes and Ana Maria Antunes, to my girlfriend Catarina, and to my very best friend Tiago.

Abstract

Over the years, collaboration between humans and organizations have been increasing and becoming vital to face new challenges and achieve the greatest common goals.

The development of new technologies and internet capabilities promoted the emergence of new collaboration types, i.e., collaboration using software connected through internet (Collaborative Workspaces software). The use of the internet amplifies the range of action and the speed of communication among the actors involved in a collaboration.

The collaboration amongst organizations is project-oriented (the common goal is to deal with projects) where several actors involved in the collaboration share their knowledge with each other. These actors are, indeed, the knowledge holders and the system which supports the collaboration has to collect and assess the knowledge from them. For this reason, this thesis aims to design and implement a software infrastructure to capture and capitalize the knowledge created over several projects.

Such software is human-centered and has an autonomous, proactive and reactive behaviour to handle all users' needs. This software promotes its own continuous learning by analysing humans' behaviour over several projects, extracting information from that behaviour, and having *Context-awareness*. Additionally, it relies on Data mining technologies and semantic services, in order to provide a continuous monitoring of the whole project during its life cycle.

The software developed is called *"Companion"* and has been assessed as a part of the CoSpaces Integrated Project.

Resumo

Ao longo dos anos, a colaboração entre humanos e organizações tem vindo a aumentar e a tornar-se vital para enfrentar novos desafios e alcançar os grandes objectivos comuns.

O desenvolvimento de novas tecnologias e capacidades da Internet fomentou o aparecimento de novos tipos de colaboração, ou seja, softwares de apoio à colaboração ligados através da internet. O uso da internet amplia o raio de acção e a velocidade de comunicação entre todos os actores envolvidos numa colaboração.

A colaboração entre as organizações é orientada para projectos (o objectivo comum é lidar com projectos) onde os vários agentes envolvidos na colaboração partilham o seu conhecimento uns com os outros. Na verdade, estes actores são os portadores do conhecimento e o sistema que suporta a colaboração tem de recolher e avaliar o seu conhecimento. Por isso, esta tese tem como objectivo projectar e implementar uma infra-estrutura de software capaz apreender e capitalizar o conhecimento obtido ao longo de diversos projectos.

Este software possuiu um comportamento autónomo, pró-activo e reactivo capaz de atender a todas necessidades dos utilizadores (human-centered). Tal software promove a sua própria aprendizagem de forma contínua e gradual, através da análise do comportamento dos seres humanos ao longo de vários projectos. Adicionalmente, baseia-se em serviços de mineração de dados e serviços semânticos, para fornecer uma contínua monitorização de todo o ciclo de vida de um projecto.

O nome do software desenvolvido é "*Companion*" e foi avaliado com parte do projecto *CoSpaces*.

Acronyms List

BI – Business Intelligence

OLAP - Online Analytical Processing

KM – Knowledge Management

JADE - Java Agent DEvelopment Framework

WBI - Web Business Intelligence

KS – Knowledge Support

K-Elems – Knowledge Elements

CoSKS – CoSpaces Knowledge Support

CSF – CoSpaces Software Framework

FSC - Full Services Configuration

IMSC - Integrated Miner Services Configuration

XML - eXtensible Markup Language

BSCW - Basic Smart Cooperate Worldwide

IDE - Integrated Development Environment

ORM - Object-Relational Mapping

OSWAF - Open-Source Web Application Framework

RDBMS - Relational Database Management System

JSP - Java Server Pages

UML - Unified Modeling Language

JVM – Java Virtual Machine

SQL - Structured Query Language

MCV - Model-View-Controller

API - Application Programming Interface

ERD – Entity-Relationship Diagram

UML - Unified Modelling Language

ICE - Interface, Control and Entity

Figures List

Tables List

Index

Chapter 1 - Introduction

1.1 Rationale

Collaboration among humans have always been vital to achieve some of the greatest conquers in human history. In fact collaboration is the oldest and simplest method of transferring and sharing knowledge among humans. Transferring and sharing knowledge allow a constant evolution and improvement of humans' capabilities, competitiveness, and decisions. However, "trust" is the key felling that enables and support the knowledge sharing among people, so it is necessary to create good collaboration conditions to promote the trust and the knowledge sharing.

Collaboration can be face-to-face, where people interact with each other in the same place at the same time or can be a remote collaboration, where people interact with each other using internet, telephone, e-mail, and they are geographically dispersed.

In the engineering universe, projects are the classical way of collaboration where geographically dispersed partners work together to achieve a common goal. Such a level of common understanding enables an effective and efficient collaboration among different partners. In fact people are the knowledge holders, who possess the relevant knowledge, and by working together in many projects they are able to leverage their knowledge by sharing and combining experiences with each other.

Knowledge is vital to capitalize and can be used as a key conveyor of business strategies, by aiming to improve organization capability, efficiency, and response. Sharing knowledge among project participants amplifies the knowledge accessibility and enhances the performance and quality of all decisions made during the project's life cycle. Nevertheless it is important to have some techniques to properly manage that knowledge.

Knowledge Management (KM) comprises a collection of strategies and practices to identify and enhance the process of creation, representation, distribution, capitalization,

and knowledge sharing, and to enable the adoption of experiences achieved. KM techniques can be used to help project participants.

Collaborative Workspaces (CW) can be used to handle the knowledge and to support its life cycle. CW are internet-based environments that intend to support the collaboration amongst people spread all over the world. Using Collaborative Workspaces project partners are able to interact with each other (e.g. sharing knowledge, projects, expediencies, etc.) through a single software or entity.

Over the years, we have witnessed a significant and sustained use of Collaborative Workspaces in several areas, which nowadays assumes a relevant role within companies' strategies. However, there is still some reluctance to use this technology due to the manager roles' division and hierarchy.

Collaborative Workspaces evolution depends on both ability to generate return on investment and the capability to enhance the performance of all participants engaged in a collaboration, reducing the time spent on projects' or tasks' life cycle.

It is important to integrate KM functionalities into CW, which are able to have "context awareness" over all running processes, analyze those processes, assess knowledge through them, make forecasts, and propose solutions to existing problems. These functionalities shall also be provided with proactive behavior, in order to detect and fulfill all needs of the users involved in collaboration and reduce their decision time.

In order to assist both CW and KM implementation, Business Intelligence (BI) refers to skills, processes, technologies, applications, and practices used to support better decision making.

The development of different Business Intelligence architectures and approaches have accelerated, improved, and facilitated all the processes existing on a CW (e.g. knowledge sharing, decision making, finding solution for problems, etc.). This fact reduced some stigmas associated with this type of collaboration and enhanced the KM techniques.

The present thesis designs and implements a reactive, proactive, and autonomous software infrastructure called *Companion* to provide the collaborative systems with KM functionalities. This software infrastructure makes the collaboration life cycle easier and Collaborative Workspaces more useful and helpful, since it provides proactive support to

the decision making process, forecasts, accurate information of all existing entities (e.g. tasks, projects, meetings, and actors), offers relevant and important documents to users, provides performance enhancement, proposes solutions for problems, and has “context awareness” over all running processes.

The *Companion* has an adaptable, open, flexible and scalable architecture, which makes it a component easily integrated to heterogeneous environments.

1.2 Goals

This work targets the development of a software infrastructure to help improving the decision making process in Collaborative Workspaces.

Such general goal can be divided into the following specific objectives:

- Observe all processes involved in a project (e.g. tasks, issues, meetings, actors selection, milestones, and deadlines) and detect possible deviations or problems that could jeopardize the project's success.

- Offer a set of solutions to solve a particular problem or a pertinent and appropriate document to help the user, always regarding the performance and efficiency enhancement.

- Provide a constant surveillance and monitoring of the work that is being carried out.

- Offer personalized information to each user, according to their features.

- Create an adaptable, open, flexible, and scalable component, in order to make it an easy pluggable component.

1.3 Context

This thesis was developed within the CoSpaces Integrated Project, funded by the European Commission. Therefore this thesis a close relation between CoSpaces needs and this thesis objective.

The overall objective of the CoSpaces Project was to develop organisational models and distributed technologies that support innovative collaborative workspaces for individuals and project teams within distributed virtual manufacturing enterprises, so as to establish effective partnerships, collaborate, be creative, improve productivity, reduce the length of design cycles and take a holistic approach to implementing product phases.

The project aimed to achieve this through enhanced human communication, innovative visualisation, knowledge support and natural interaction. The framework supporting this thesis focuses on the knowledge-related dimension of CoSpace which is handled by the CoSpaces Knowledge Component. Such a component is, indeed, a set of ontology-enabled knowledge services (named CoSKS), which provides the appropriate functionalities to support the knowledge requirements identified as part of the operational scenarios of the project.

The Companion software infrastructure is part of the CoSKS and, as such, CoSKS is introduced here, through the description of the major conceptual axes guiding its development, the operational environment (or the context of work), the requirements guiding the development of CoSKS and, finally, the CoSKS conceptual architecture including also relevant topics for this discussion (e.g. the CoSpaces ontology, the knowledge items/objects, etc.).

As previously referred, the very target of this thesis is the development of the Companion, an autonomous, proactive, and reactive component to help improving the management level of a given project.

1.4 Structure of the Document

This document is structured in six chapters. Chapter 2 describes the current state of the art of main areas involved in this work, namely Business Intelligence, Knowledge Web Business Intelligence & Collaborative Workspaces, and Context-awareness & Proactivity.

Chapter 3 presents the conceptual model supporting the work. Additionally, it also provides a set of basic definitions used here, it describes the context of the work within the CoSpaces project, and it introduces the main conceptual elements behind the Companion and their applications.

Chapter 4 describes the Companion implementation, including technologies and tools used. Since UML was adopted in the Companion development, some diagrams such as, Use Cases, Classes Diagram, and DER Diagram are also presented here.

Chapter 5 draws some considerations about the contribution of this research, main results achieved, implications for future research, and the limitations of the study.

Chapter 2 - State of the Art

This chapter summarizes the current state of the art of the main areas involved in this work, divided into six different sections. Section *2.1* introduces the main areas involved in this work, namely *Knowledge & Knowledge Management*, *Behavioral Systems*, *Business Intelligence*, and *Collaborative Workspaces* which are discussed in sections *2.2, 2.3, 2.4* and *2.5*, respectively.

2.1 Main Areas Involved

This work touches four main topics namely, *Knowledge & Knowledge Management*, *Behavioral Systems*, *Business Intelligence*, and *Collaborative Workspaces.*

Knowledge & Knowledge Management covers the management of the acquired knowledge and helping the decision making process. *Behavioral Systems* is related to systems with dynamic behavior and special behavioral features. *Business Intelligence* is a technology designed to help the decision-makers, by offering them accurate and reliable information. *Collaborative Workspaces* are virtual environments where people from geographically dispersed locations can work, interact, and share knowledge with each other.

Figure 2.1 illustrates how those areas are inter-related and how they set the borders and offer the ground for the conduction of the work proposed in this thesis. Notice that *Behavioral Systems* are integrated within *Knowledge & Knowledge Management* and for this reason its features are only shown and explained on Figure 2.2.

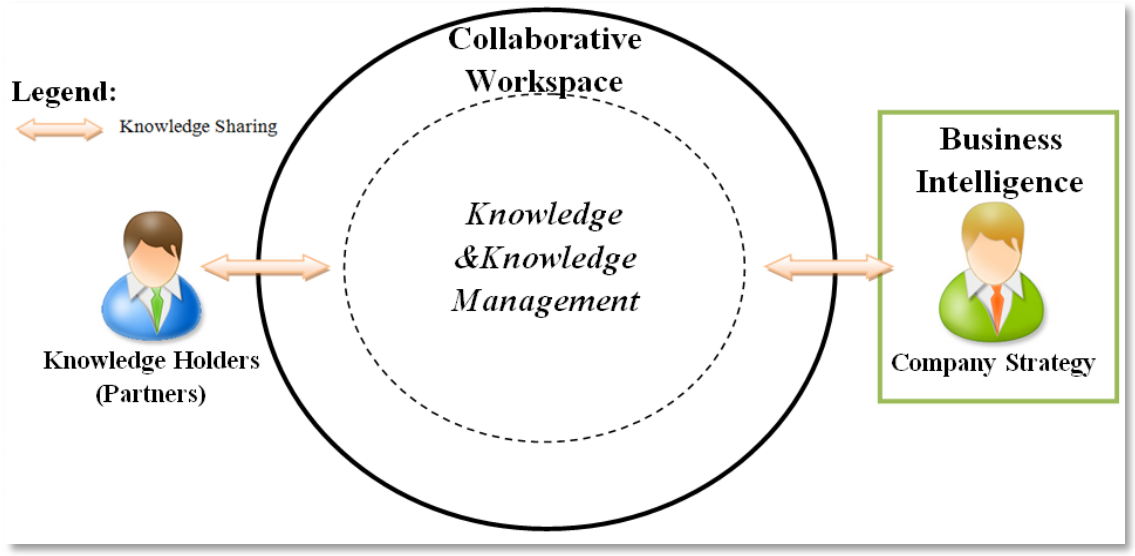


Figure 2.1 - Main Areas Involved in Knowledge Sharing

The knowledge holders use the collaborative workspace to perform projects and share their knowledge among themselves. This knowledge is transferred to Collaborative Workspaces, where it will be captured, capitalized, organized, and filtered in order to provide the project participants with updated knowledge.

It is important to refer that companies involved with a project define their own business strategy using Business Intelligence technologies, and share their knowledge using Collaborative Workspace, as well.

In the context of this work KM is seen as a type of three elements, namely *Reasoning, Behavioral Systems,* and *Semantic Services & Ontology* (Figure 2.2). Behavioral Systems are the key topic to be closely considered and explored throughout this document.

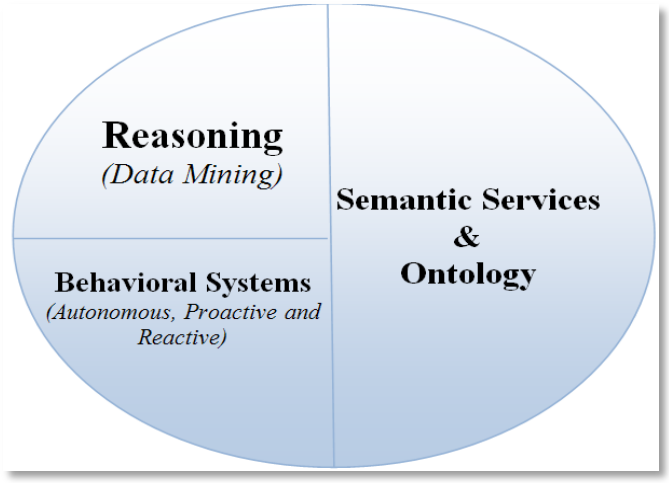


Figure 2.2 - Knowledge Management Topics

For the sake of clarity, it is important to define three basic concepts commonly referred in this area of work, namely: *Data*, *Information,* and *Knowledge.* The Table 2.1 shows those definitions.

Data	Stored information to be examined and used to make decisions.
Information	Facts or details about a specific topic, presented within a context that gives them some meaning, relevance, and increase the context understanding [1].
Knowledge	Understanding, awareness and skills acquired by a person through experience or education [1].

Table 2.1 - Basic Concepts

2.2 Knowledge & Knowledge Management

As previously stated, knowledge sharing among partners engaged in a project may potentially enhance the decision making process. Decision making process is, indeed, the process that leads companies/collaboration groups to an action. Within a project based on a collaboration environment, typically most decisions are complex, difficult to make, and interdisciplinary sometimes resulting in conflicts among project participants. Decisions require several inputs from different knowledge sources, i.e., knowledge extracted from humans or from stored data. Nevertheless, the extracted knowledge needs to be tailored to the project participants needs.

In order to handle the acquired knowledge, KM techniques are widely used and its importance has been increasing. KM is a set of processes that identify, organize, collect, and create knowledge from a collection of data, and facilitate the communication between knowledge creator and those who will use the achieved knowledge and make a decision or an action [1].

The set of tools focused on the creation and management of knowledge have three characteristics in common: *Accessible Information, Support in decision-making and Guidance to final user* [2].

Zhang and Lu [3] identify two categories of knowledge: *Quantitative Knowledge* and *Qualitative Knowledge*. The former is acquired from both internal and external resources,

i.e., knowledge which is extracted from data (Data-driven knowledge). The later is achieved from collaboration with other people, knowledge created by an expert (Human-drive knowledge).

Broadly speaking, whilst quantitative knowledge provides better understanding of business/collaboration processes and more reliable decisions for decision-makers, qualitative knowledge is based on human's experiences, in order to make decisions quickly and promptly.

Nonaka and Takeuchi [1] classify tow different types of knowledge: *Explicit Knowledge* and *Tacit Knowledge. Explicit Knowledge* is described in details, formal, systematic and can be expressed in words and numbers. *Tacit Knowledge* is described as a knowledge built on experiences, and it is difficult to capture and share, because it is highly personal and not easily visible or expressed.

There are four processes of knowledge conversion involving those two types. [4]:

1) *Socialization*: From tacit knowledge to tacit knowledge.
2) *Externalization*: From tacit knowledge to explicit knowledge.
3) *Combination*: From explicit knowledge to explicit knowledge.
4) *Internalization*: From explicit knowledge to tacit knowledge.

These processes are described in the SECI model proposed by Nonaka and Takeuchi (Figure 2.3).

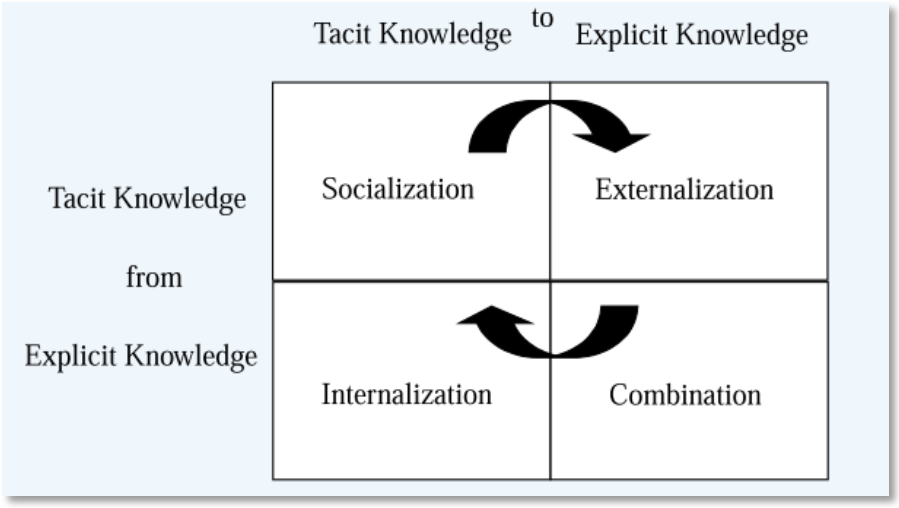


Figure 2.3 - Nonaka and Takeuchi model of knowledge conversion [4]

Socialization (tacit-to-tacit) happens through direct interaction of people, when tacit knowledge is freely exchanged. It is an experimental and active process that involves capturing and sharing knowledge in face-to-face and real-life situations.

Externalization (tacit-to-explicit) makes tacit knowledge explicit, i.e., transforms tacit knowledge into reliable and understanding knowledge. This process occurs when a person (or group) converts ideas, experiences or images into words, concepts or analogies. Once externalized, knowledge can be easily shared with others.

Combination (explicit-to-explicit) happens when discrete pieces of explicit knowledge are joined into a new form of explicit knowledge. Once we have different fragments of explicit knowledge, it is possible to convert and combine them into documents, email, databases, etc.. The *combination* makes the explicit knowledge more usable and transferable among groups across organization or collaboration.

Internalization (explicit-to-tacit) is the process for comprehending and absorbing explicit knowledge into tacit knowledge, in order to create or improve the mental knowledge model of individuals. This process transfers explicit knowledge to an individual worker, helping the workers to internalize the achieved knowledge from previous experiences.

Nonaka and Takeuchi developed a knowledge spiral model, arguing that the knowledge creation process is a dynamic and continuous flow between tacit and explicit knowledge.
Figure 2.4 shows the knowledge 3D spiral model where knowledge continuously flows in an evolving process that involves individual, groups of people, organization and groups of organizations.

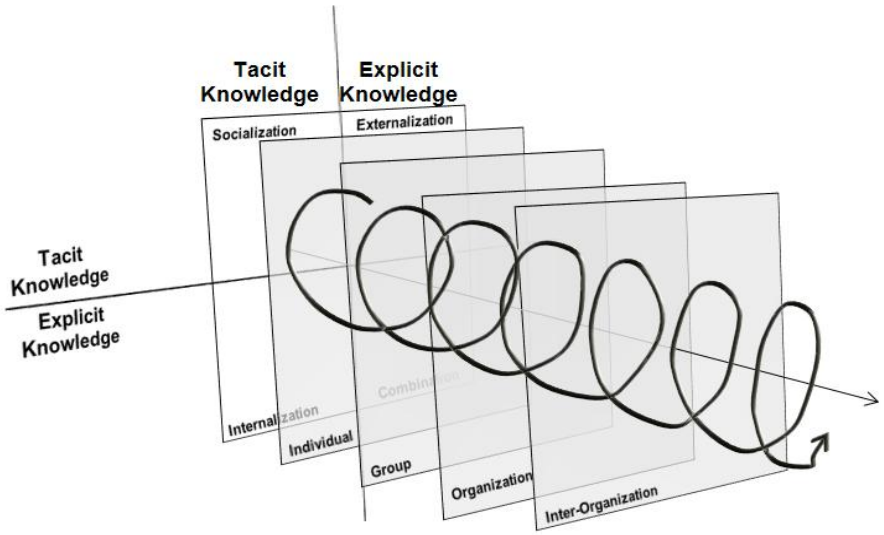


Figure 2.4 - Nonaka and Takeuchi knowledge 3D spiral [3]

The knowledge spiral model represents the knowledge transfer process as a spiral, in order to clearly show that knowledge evolves during the transfer process. It starts with tacit knowledge (socialization) that is externalized in the process of attempting to articulate it to someone else. The externalized knowledge is combined and internalized with the existing knowledge.

Relevant literature shows different KM models. However, *Kimiz Dalkir* [4] proposed an integrated model supported by the knowledge spiral model from Nonaka, which is structured in three phases: 1) *Knowledge Capture and/or Creation; 2) Knowledge Sharing and Dissemination; 3) Knowledge Acquisition and Application.* Figure 2.5 shows the interactions between those three phases.

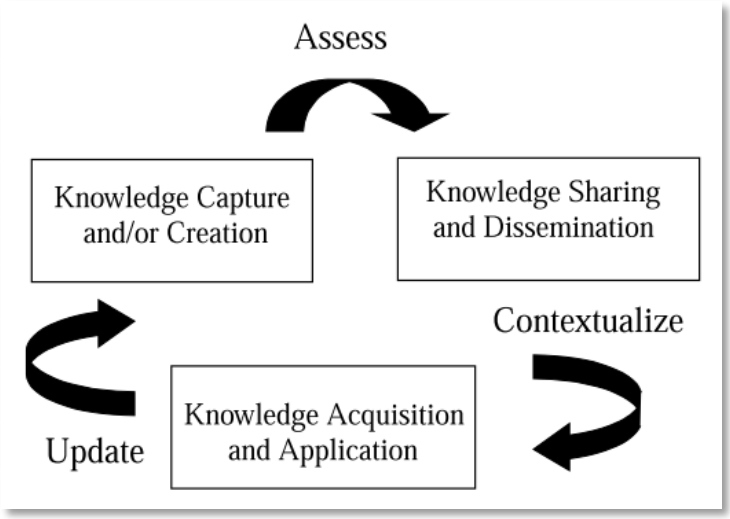


Figure 2.5 - Integrated Knowledge Management model [4]

1) *Knowledge Capture and/or Creation*

In order to be used, knowledge competencies and experiences must be registered and formalized. The Knowledge Capture phase acquires and filters the captured knowledge, verifying its quality and preparing it for future storage.

2) *Knowledge Sharing and Dissemination*

Knowledge Sharing and Dissemination goal is to access and disseminate existing knowledge. Sharing knowledge among organizational workers or collaboration participants is crucial to develop norms of trust, reciprocity and cooperation, thus knowledge sharing enhances organization or collaboration efficiency. However, it is crucial to guarantee that

knowledge is available to every worker inside an organization, who must be provided with updated knowledge.

3) Knowledge Acquisition and Application.

In this phase knowledge is used and applied to support decisions in order to increase company/collaboration benefits. To increase the knowledge management gain, is important to guarantee the application of relevant and trusted knowledge in real and adequate situations or problems.

2.3 Behavioral Systems

Behavioral Systems enrich software with dynamic behavior and specific behavioral features. With Behavioral Systems it is possible to create self-learning and self-adapting software in order to handle environmental changes. There are many types of behavior such as: *conscious or subconscious, overt or covert, voluntary* or *involuntary*, *proactive, reactive and autonomous.* Tools and frameworks like JADE or IBM Classification Module, provide software with dynamic behavior *(self-learning and self-adapting*), however build a proactive and autonomous software is still a challenge.

In the Collaborative Workspaces context, Behavioral Systems handles two main behaviors: *Context-awareness* (autonomy) and *proactivity*. With *context-awareness* the software is able to know and detect real-time changes, adapting its role and collecting important information to use in further problems solving. Thus, users' roles and permissions must be constantly adapted regarding changes of context and their intentions. However, previous research has found some difficulties to describe the main conditions for assigning roles, intentions and modifying permissions [5].

Since environmental changes might occur at any time, system definitions and requirements may become temporally obsolete, resulting in wrong information response [6]. In order to deal with this problem, reliable and fast context detection and adaptation is highly relevant to the decision-making process and to enhance the user interaction and experiences.

Context-awareness development usually flows into two ways: *Self-supported context-awareness* and *infrastructure-support context-awareness.* In *self-supported context-*

awareness, software has its own mechanisms and technologies to perceive the environment and act according to that. In *infrastructure-support context-awareness*, software achieves the context changes through an external hardware and software, associated with it [7].

It is possible to create a collaborative workspace with context-awareness capability, by exploring different technologies (e.g. virtual reality, sensors, voice microphone, ambient-noise, temperature sensors, light sensors, and cameras).

Proactive tools are essential to increase users' efficiency within collaborative workspaces environment, and to reduce time needed to find information and make proper actions. Through previous and relevant information and actions, those tools are able to supply project manager and participants with important files, reports, information about previous projects, and best practices. Instead of spending time in document searching and finding the best way of solving a problem, with proactive tools the user gets fundamental information and decisions to help him to solve the problems. Those tools increase the efficiency and performance of collaborative workspaces, by providing the users with possible solutions for problems and relevant/important documents related with tasks that are being performed.

Even so, it is particularly important to refer that a proactive tool does not make any decision without a "human" intervention. A proactive tool is just a decision-helper and not a final decision-maker.

2.4 Business Intelligence

Business Intelligence (BI) concept emerged in 1996 through the Gartner Group [8]. BI is usually described as a technology designed to help decision-makers by giving them accurate and reliable information about their corporation [8, 9]. Based on the organizations' best practices, BI enables better decisions, actions and business processes. BI processes improve the knowledge and value of decision-makers, since it offers a range of organized and filtered data. With all processed information, achieved through BI technologies, administrators, managers, and employees can make decisions and execute the appropriate actions to solve their issues.

In fact, Business Intelligence tries to improve decision-making process and solutions for problems, in order to reduce the costs and the data warehouses of the companies [5]. With the fast market and organizations' expansion and growth, companies across the world needed appropriate decision and knowledge support to face their new challenges.

BI mainly uses modern applications and information technology, such as Data Warehouse, Online Analytical Processing (OLAP), Data Mining, queries, and reports [2] in order to collect, manage, store, and analyze structured and unstructured data. BI applications could be divided in two categories, namely: *Information Application and Knowledge Application* [11]. *Information Application* processes original data and transform it into a set of acceptable and feasible information for decision-makers, focused on data queries, reports & charts, multi-dimensional analyze, and data visualization [11]. *Knowledge Application* refers to Data Mining technology and Data Warehouse to find implicit relation of data and to form and represent reliable and comprehensive knowledge for decision-maker.

Data warehousing represents the filtration, organization, storage, analysis, interpretation and consolidation of data. Typically, Data warehousing also includes data-mining modules and decision support modules [12] which enhance system's predictions and adaptability.

The main drawback of Data Warehouse is its time-consuming process of collecting and transforming data. Thus, it causes time delays between the recognition of an event and its analysis, creating data latency and reducing the system efficiency. Data latency is frequently described by the time it takes to inform (either the system or the person in charge of data analysis) that new data is available to be analyzed [13].

Data Mining collects and identifies trends and behaviors in order to produce important information for the decision-makers. Through the combination of powerful Data Mining techniques with Data Warehouse, it is possible to increase the size of the knowledge repository. Data mining is a flexible approach which allows the application of different data exploration techniques. Data Mining techniques could use efficient algorithms and powerful tools to increase its forecasts, knowledge, solutions, and reasoning, reducing the time consuming and data analysis latency [13]. Data Mining can also adapt its mining techniques for a given problem, aiming to avoid the time required to perform a given analysis (Latency problems).

OLAP is an important application used in BI which reduces the time needed for analysis and data's transformation, providing a powerful tool to help data transformation. However OLAP needs a fixed dimensional structure and that could be a problem in dynamic and flexible systems [13].

As mentioned before, latency problems are, indeed, an important concern in Business Intelligence research. BI handles a set of requirements to create, manage and enhance decision-making process and give flexibility and agility to an organization. This flexibility and agility increases the organization dynamic, natural competition, and fast response to new challenges and environmental changes, creating a high differentiation between organizations. It is important to reduce the gap of time between the occurrence of a business opportunity and organization action time. By reducing this gap, an organization can be ahead of other organizations and increase the likelihood of its success.

Figure 2.6 shows that the more action time an organization spends the less business value it gets, so a short action time is needed to get the highest business value from a business opportunity. Action time refers to three components: *data latency, analysis latency, decision latency.* [13]. *Data latency* is the time taken to collect and store the data. *Analysis latency* is the time needed to analyze the data. *Decision latency* is the time needed to make a decision.

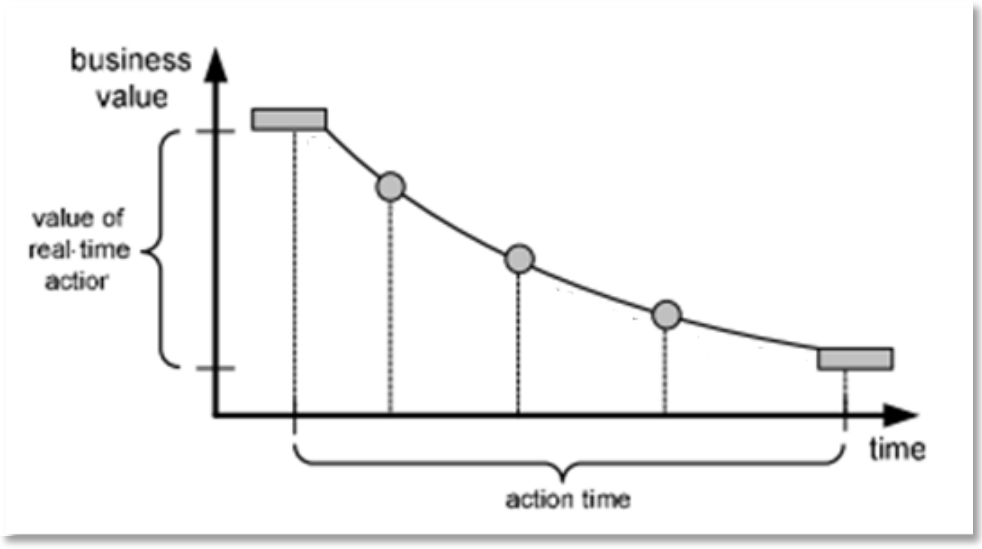


Figure 2.6 - Business value and reduced action time [13]

2.5 Collaborative Workspaces

Project-based collaboration gained a new dimension within the Internet era. The sharing and capitalisation of knowledge within a project can be highly amplified through the use of Internet services, since it can reach a bigger audience in a more effective way.

With the exponential raising and development of internet, most organizations and companies have moved their approach to e-business where the internet is the main platform to interact with their customers. Collaborative Workspaces concept has emerged in order to handle this new challenge. *Collaborative workspaces* are virtual environments in which all the participants or teams, from geographically dispersed locations, can access and work in a common project and interact with each other through a single software/workspace.

Working on virtual and collaborative workspaces is a common reality among engineers and, more and more tools and software for this purpose have been developed (e.g. *Microsoft Groove, IBM Lotus Notes, IBM Workplace, and e-Office*). Even though, provide users with accurate information, important decisions, forecasts, have a proactive behavior, and context-awareness, is still uncommon on these software category.

Tools supporting collaborative workspace shall be adaptable, scalable, improve collaboration and productivity, make users/works' communication easier and provide venues, forecasts, and solution for problem solving. To meet all these requirements it is important to have a software framework built on advanced web services, context modeling technology, and context-awareness to support the collaboration life cycle [14].

When integrated with Data Mining techniques, Collaborative Workspaces appear as one of the most powerful and useful technology to handle with multidimensional and geographically dispersed teams.

Chapter 3 - Requirements & Conceptual Model

This chapter explains the conceptual model that supported this thesis. It presents the basic terms definitions adopted in this work, the Conceptual Foundations of the work, the Decisional Gate Supporting Process, CoSKS Technical Foundations, Behavioral Capabilities within *Companion* (*Autonomous, Proactive* and *Reactive* behavior), the *Companion's* Interactions, and the *Companion* Behavioral Services.

3.1 Basic Terms Definitions

For the sake of clarity, this section provides a short definition of the basic terms used in this thesis (Table 3.1).

Term	Definition
Project	It is a collection of linked tasks, carried out by actors/participants engaged in collaboration, with a defined start point and end point. The main goal of this "linked tasks" is to achieve the results needed and expected by an organization or collaboration.
Meeting	It is an electronic or face-to-face conference among participants engaged in a project. Meetings are associated with a list of agendas which have specific and defined dates, duration and local.
Task	It is an activity that must be completed within a defined period of time, under responsible supervision. Tasks are carried out by actors and have a defined start date and end date.
Issue	It is a known deviation or problem that has happened and must be solved promptly. To solve an issue, a set of tasks are created by a project manager.
Agenda	It is a topic to be discussed in a specific meeting. Each topic concerns a specific issue. All the Agendas are associated with a meeting.

Decisional Gate	It is a control point at some clearly defined stage in the project. This control point starts with a set of inputs (topics to be discussed) and produces a set of outputs.
Report	It is a document which compiles information about a specific project, task, meeting or issue. This information can be: **a)** A highlight of project's progress using Decisional Gates; **b)** A concise summary of ongoing project, task, meeting or issue; **c)** Information about finished relevant and similar projects, tasks, meetings or issues.
Deviation	It is any uncertain or unplanned event that occurs during the project or task which can jeopardize its success.
Project Manager	It is the main responsible for the project work and success from its beginning until its closure. He or she is the key decision-maker of the project, i.e., select project's team, allocate and monitor resources, solve problems, track project's progress, etc..
Actor	It is someone who is involved in a project and has a specific role.
Role	It is the function or position that an actor or participant has or is expected to have in a specific activity.
Behavior	It is an action or reaction of a person or component in relation to environment or other person.
Autonomy	It is a feature of a system or component capable of self-management. It is the capability of making decisions by its own and using high-level policies, adapting itself to any environmental changes.
Context Awareness	It is the ability to be aware of what is happening around the system and to understand what that information means to the system.
Knowledge Element	It is a piece of knowledge focused on specific topics. It may include documents about lessons learnt, best practice, expert manuals and expert contact information. In the CoSKS context projects, templates, agendas, meetings, tasks, and issues are also considered knowledge elements.
Collaborative Workspace	It is a web-connected environment in which all the participants in dispersed locations can access and interact with each other just as inside a single entity.
Companion	It is a proactive, reactive, and autonomous software infrastructure which enables interactions between users and the system, managing alerts, tracking project deviations and supporting the decision making process

Table 3.1- Basic terms definitions

3.2 CoSpaces Project Requirements

As already mentioned, the CoSpaces Project settled the context of development for this thesis. As such, this work targeted some CoSpaces needs, which are described in this section.

Knowledge dimension of CoSpaces is handled by the *CoSpaces Knowledge Support* component (CoSKS). CoSKS is part of the CoSpaces Software Framework (CSF), which is a collaborative engineering environment supporting real-time collaboration between geographically dispersed teams. The CSF uses advanced technologies including virtual reality, augmented reality, tele-immersive interfaces, mobile technologies, context awareness and web services, to create human-centric collaborative workspaces. The knowledge-related functionalities are to be ontology-enabled and combined with the data mining techniques in order to fully support the development of projects.

The CoSKS approach focuses two major dimensions: issues and decisions. Such approach intends to keep track of all the issues occurred during the project associated with the decisions that were made in order to solve those issues. The bottom line is to provide the project teams with a better understanding of common problems which occur frequently in projects and also the strategies (tasks and actions) that were carried out to overcome them. The goal is to improve the daily work of project teams by providing them with historical cases from previous projects and how these cases were solved, thus reducing the time looking for good options in specific situations. This can be done when a new project is about to be started (e.g., the project manager can get relevant knowledge about successful projects in the same area, such as deviations, major problems x solutions, etc.) or during the meetings themselves (participants could have a more accurate knowledge about the project, by knowing the risks associated to a given decision, considering the existing knowledge). Moreover, CoSKS also needs to keep track of important decisions made during the meetings, through the creation of template-based agendas (including topics to be discussed, participants, etc.) and minutes (issues raised, solutions proposed, and decisions made).

The figure 3.1 shows the CSF architecture and highlights the relevant and important points for CoSKS. From the CoSKS perspective (label ❷), two components are very relevant, namely the CoSpaces Portal (label ❶) and the BSCW. BSCW (Basic Support for

Cooperative Work) is a document management tool which is the place or repository where Knowledge Elements *(K-Elems)* will be stored or retrieved.

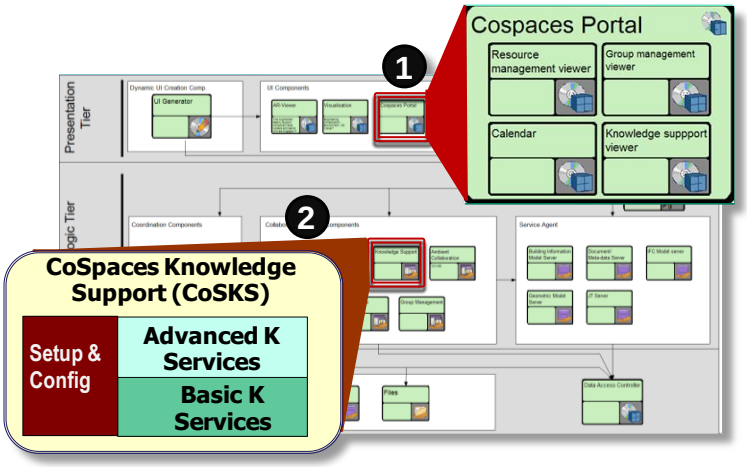


Figure 3.1 - The CoSKS within the CSF architecture [20]

K-Elem represents pieces of knowledge that can be captured, stored, published, shared, and reused among the project teams. K-Elem is the relevant knowledge to provide the proper support to e-collaboration in a given project. In addition to ordinary documents, some specific examples of K-Elems used here are: **project, issues, solutions, agendas, minutes, tasks, participants,** and **project** post-mortem. The K-Elems will strongly rely on ontological concepts, especially in terms of indexation and classification for future search and reuse.

Throughout the CoSpaces Portal the users (manager and ordinary users) might have access to the knowledge-related functionalities provided by CoSKS, through the element named *Knowledge Support Viewer* (label ❶), which is a kind of *knowledge entry door*. Such a component provides management and exploitation of the knowledge produced in the CoSpaces collaborative environment.

The CoSpaces Knowledge Support (label ❷) is organized conceptually in three main parts, namely:

- ***Setup & Config***: handles the needs related to setup and configuration of the CoSKS component (e.g., creation of users/passwords, servers' addresses, etc.);

- ***Basic Knowledge-Services***: the vital services for the normal operation of the CoSKS. Without them, the CoSKS cannot work properly. Some examples are: Knowledge Elements indexation, ontology-based search, calculation of semantic weights, browsing of CoSpaces ontology, store / retrieve Knowledge Elements, etc.; and

- ***Advanced Knowledge-Services***: intended to extend and enrich the CoSKS capabilities. They are built upon the Basic Knowledge-Services. These services can be considered a high level of knowledge services. Some examples are: management of "context awareness", historical analysis of similar and relevant projects or tasks, lessons learned advisor, assesses knowledge through the processes, to forecast and propose solutions to possible or existing problems. The software infrastructure developed in this thesis is a part of these ***Advanced Knowledge-Services***.

The knowledge support provided by CoSKS, during collaboration process, can occur either in an isolated way or as part of the development of a long project. Additionally, other requirements must be considered by CoSKS. Those requirements are shown in figure 3.2 categorized into three classes: ***Functional, Architectural and Technical*** requirements.

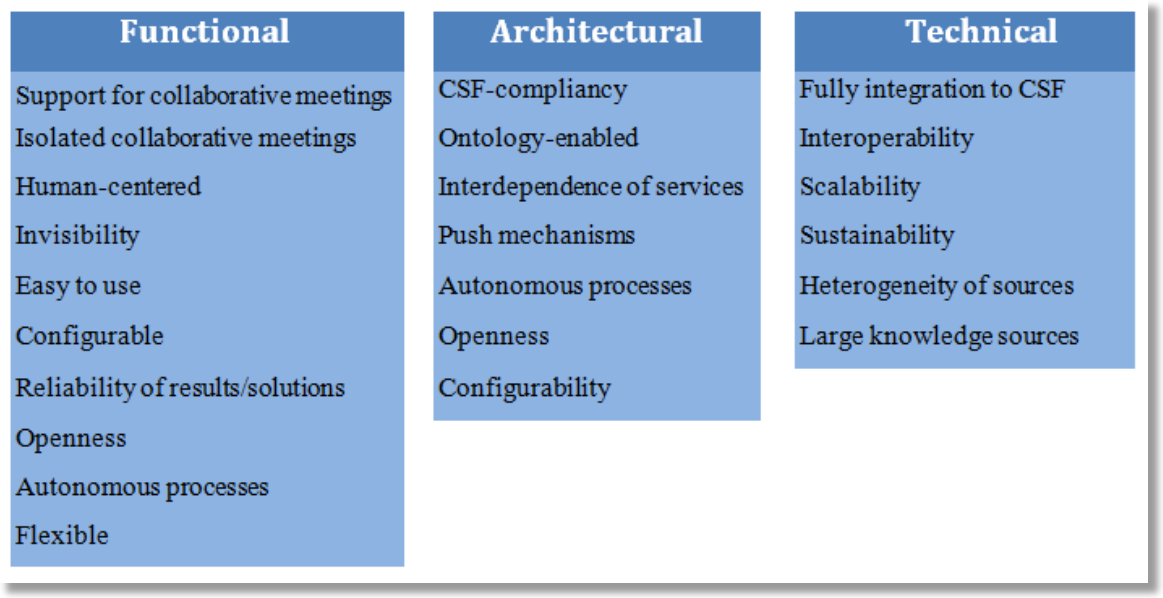


Figure 3.2 - CoSKS main requirements

All these requirements were taken into account in this thesis, nevertheless *Human-centered, Push mechanism,* and *Autonomous processes* are the core requirements. *Human-*

centered is something that interacts with people in an intelligent and cooperative manner; Push *mechanism* is a way of pushing the information to the user who would be interested in that, and *Autonomous processes* are processes able to execute a number of basic steps without human intervention in order to achieve a given goal and make proper decisions.

3.3 Conceptual Foundations

The conceptual foundations of the work presented here are grounded on *Collaboration*, *Knowledge*, and *Reasoning* (figure 3.3). *Collaboration* is related to the work performed in context of development of internet-supported projects. *Knowledge* is the 'currency' exchanged among partners collaborating within the project. *Reasoning*, represented by the use of data mining techniques, guarantees that knowledge generated during each project must be captured, transformed, and mined in order to be better reused in future projects.

Collaboration among project members can only be established when win-win situations are created, i.e., when professionals can benefit from each other's experience and knowledge towards the achievement of a common goal. This mutual benefit can be improved if the knowledge required to perform the project is properly managed through appropriate KM tools. Additionally, the use of knowledge can be potentially maximised and provided in an agile way if appropriate reasoning mechanisms are applied to identify/discover and prepare it, in order to help publishing, sharing, use, and production of new knowledge. The reasoning mechanisms can be supported by context-aware services which are essential to reduce the complexity of shared multi-users engineering environments. Such mechanisms enable users to concentrate on their tasks by providing the task-specific information in the actual individual, group or process context and by filtering the noise of unrelated status and activity information produced in distributed cooperative work sessions.

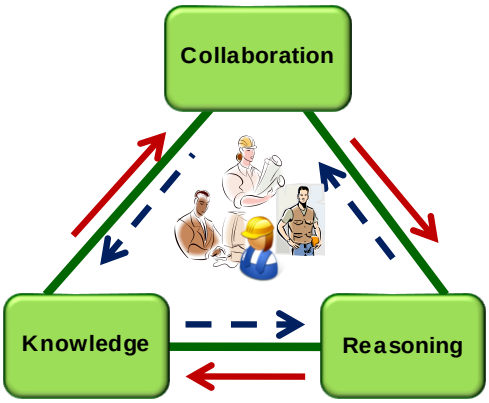


Figure 3.3 - Conceptual Foundations of the work [14]

The core idea of the conceptual foundations of this work is that projects are conducted through a series of meetings and every meeting is considered a decisional gate, a driving point where issues are raised and decisions are made in order to find the appropriate solutions to those issues (figure 3.4). Such decisions can, eventually, be transformed into tasks to be performed.

Each decisional gate is prepared (through the creation of agendas), and the events that occur during the meeting shall be recorded. Between two decisional gates there is a permanent monitoring on the execution of all tasks executed. After meeting closure, there is a need for a mechanism to enable the preparation the minutes easily, by highlighting the major decisions that were made during the meeting [14].

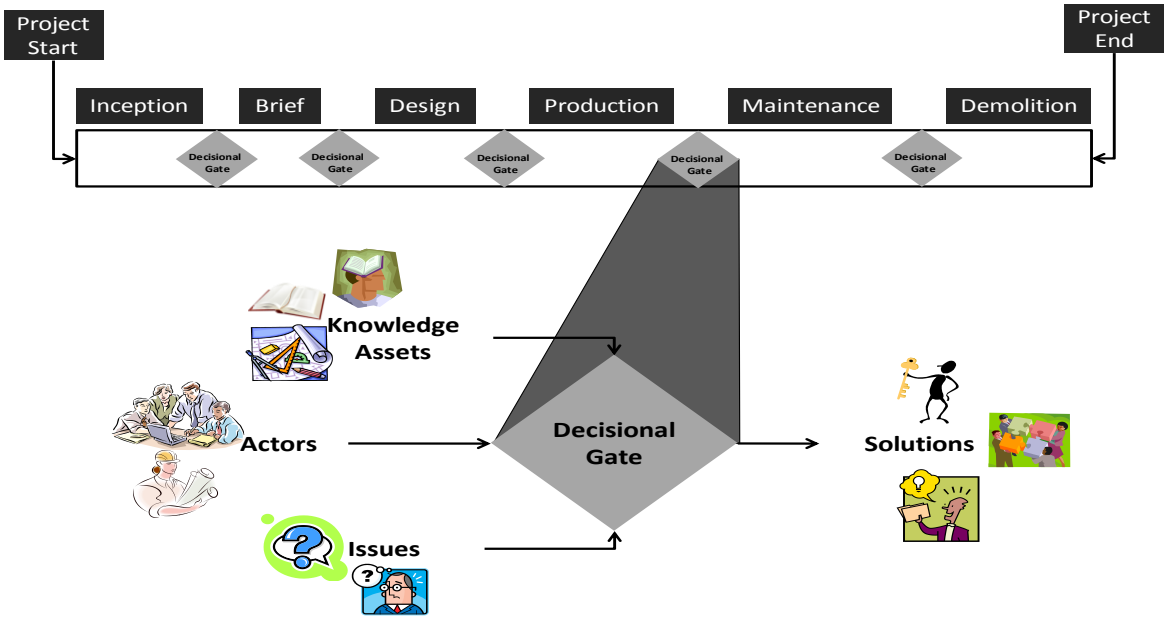


Figure 3.4 - Project decisional gates [14]

The development of a platform providing applicable knowledge depending on the user's context demands the definition of *context*. It is easily understood that *experts* (from different areas of expertise) working collaboratively in a given product have different needs/visions about/on the knowledge used, which are strongly influenced by their backgrounds, roles, responsibilities, etc.. Additionally, different types of *projects* can give different uses to the same knowledge (e.g. knowledge about accessibility regulations used in public *versus* private project buildings). Going further, knowledge can be treated differently depending on the *meeting* it is captured/used (e.g. deviations and delays in different phases of the project have highly different meanings). In different *tasks* the same knowledge may have different uses. The *issue* to be solved also defines the relevance of a given knowledge. All the previously terms written in *italic* compose the preliminary list of valid *contexts* considered here.

3.4 The Decisional Gate Supporting Process

Sessions of active collaboration generally follow a defined workflow. Figure 3.5 shows how decisional gates are structured in this work [14].

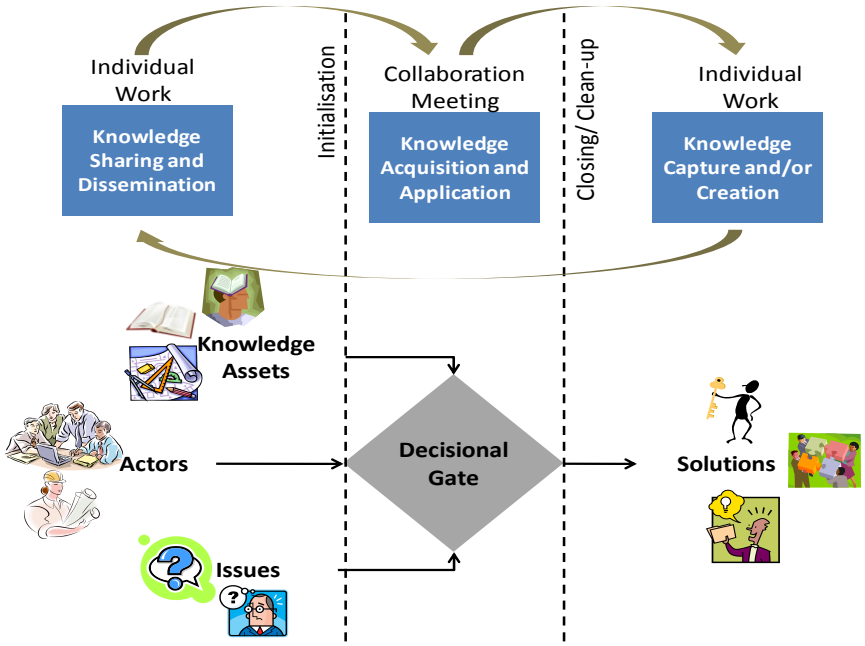


Figure 3.5 - KM process in the context of decisional gates [14]

Individual Work phase covers the identification of an engineering problem, which comprehends a description of the problem using natural language and relates to asynchronous collaboration, where all individuals involved in the project are supposed to provide inputs to the undergoing tasks.

Initialisation phase (pre-meeting) relates to the preparation of the meeting agenda and the selection of the meeting participants. The agenda creation is a semi-automated process, where a set of pre-existing templates are used to ease the process. The meeting organiser (e.g. the project manager) may (easily) identify which are the major issues/problems to be discussed in the meeting as well as the best way to present and discuss them.

Collaboration/Meeting phase is the meeting itself where participants try to reach a common understanding regarding the issues from the agenda, using the right resources. This phase also considers the annotation of the decisions made during the meeting. Those decisions, including new tasks and new assignments of responsibilities, are to be captured and meeting participants must be supported in the decision making process, according to the context of the issue discussed.

The *Closing/Clean-up* phase (post-meeting) basically targets the creation of meeting minutes. It can be supported by a semi-automated process, which provides the meeting organiser with a set of conclusions based on the decisions made during the meeting. This process may also use a set of pre-existing templates as the basis for the meeting minutes.

When the project finishes, the project manager should be able to generate a project summary (post-mortem) including all relevant indicators (issues/decisions, deviations, staff involved, etc.) that were collected during the project life-cycle. Such summary represents a K-Elem that can be reused in future projects helping to enhance performances, replicate good practices, and avoid known mistakes.

3.5 CoSKS Technical Foundations

The CoSKS is structured into four layers (figure 3.6), namely: Presentation, Behavioral, Service, and Knowledge.

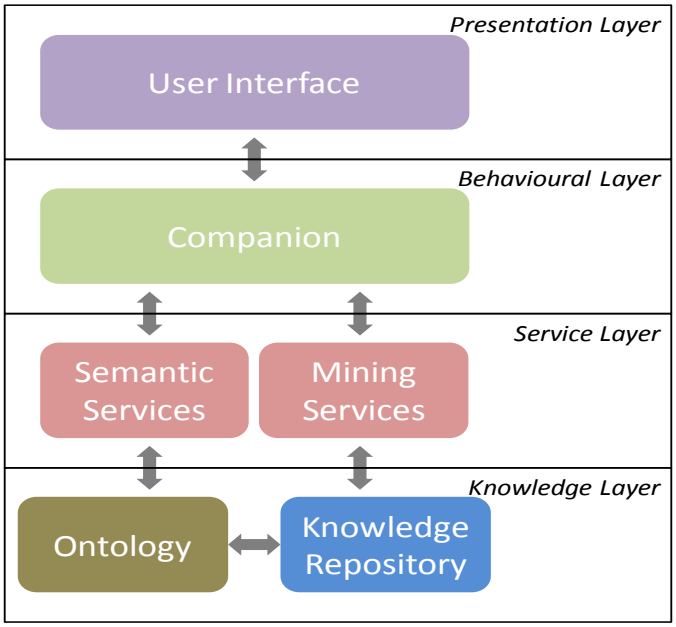


Figure 3.6 - CoSKS technical building blocks [14]

In the Presentation layer, the *User Interface* supports the interaction with the CoSKS user, through the implementation of the CoSKS portal, which symbolizes the collaboration workspace environment where the users exchange and leverage knowledge, and update the status and results of their tasks.

Companion component, as previously described, implements both proactive and reactive behaviors of CoSKS.

Mining Services provide CoSKS with reasoning-related capabilities. Among other things, they feed the Companion with accurate information in the sense that they are used to discover useful knowledge aiming to identify patterns of problems and solutions and establish the relationships between them. These services are used to anticipate problems and find potential solutions to overcome them. The main capabilities provided by the mining services are:

- **Classification:** arranges the data into predefined groups;
- **Clustering**: finding and grouping facts not previously known;
- **Regression:** attempts to find a function which models the data with the least error;
- **Association:** looking for patterns where one event is connected to another event;

- **Forecasting**: discovering patterns in data that can lead to reasonable predictions about the future.

S*emantic services* provide the following functionalities to capitalise on the ontology supporting the CoSKS operation:

- *Semantic Context and Filtering*: every K-Elem is represented by a set of keywords, which is semantically filtered in order to define its *Semantic Context*. A semantic context is built using many *elements of reference* related to the given K-Elem, such as issue, problem, actor skill, project type and phase, etc.. The semantic context behind a given query will determine the accuracy of the results provided. For instance, the same query launched by two experts (an architect and an engineer) can produce different results since their skills (and consequently, their interests) can be different when looking at the same issue (same query).

- *Semantic vector creation*: every K-Elem is represented by a semantic vector, which is a collection of the most relevant ontological concepts that semantically express it. Semantic weights are assigned to the ontological concepts in the semantic vector according to the K-Elem relevance within a given context. The semantic vector is used to classify, compare, and retrieve K-Elems that are used to provide the best possible solution (the result) for a given issue (the query).

- *Semantic indexation and retrieval*: new knowledge items are semantically indexed through the creation of their respective semantic vector. New issue/problem is issued to the system and taking into account that a query is also handled by the system as a K-Elem, it gets the respective semantic vector which is used to find similar semantic vectors of other K-Elems providing an answer/solution to the issue/problem.

The CoSKS *ontology* is developed to support and manage the use of expressions which contextualize a K-Elem within the knowledge repository. The ontology adds a semantic weight to relations among K-Elems stored into the knowledge repository. Every ontological concept has a list of ‘equivalent terms’ that can be used to semantically represent such concept. These terms are, indeed, treated in both statistical and semantic way to create the semantic vector that properly indexes a given K-Elem.

Knowledge Repository stores all knowledge elements used in CoSKS (projects, templates, agendas, meetings, tasks, issues, etc.).

3.6 Behavioral Capabilities within *Companion*

Many researches revealed that ***Behavior*** is increasingly recognized as a key component in business intelligence and problem-solving [16]. Literature shows a long list of definitions for Behavior. The Oxford Dictionary describes behavior as "*The way that somebody behaves, especially towards other people*" [15]. For this work behavior is defined as "*The way that the system behaves towards user's stimulus and inputs"*.

An important question took into account on this thesis was: *When and why does the system's behavior make difference on Collaborative Workspaces?*

Firstly, in many cases, system behavior acts as internal driving force to organization or collaboration success, making the decisions fastest and helping in problems solving. By understanding and investigating actors' behavior within project scenario and acting according to it, the system is able to enhance actors' performance and help the management level of projects.

Secondly, due to the geographic team's dispersion on a Collaborative Workspace, a system with an adequate behavior (proactive or reactive) for the project needs could make the collaboration achievable. The main idea is to increase the productivity or welfare of a user working on a CW.

Despite of existing various behavior types (e.g. *conscious or subconscious, overt or covert, and voluntary, involuntary, proactive, reactive, and autonomous*) in this study we focused on three of them, ***Autonomous***, ***Proactive*** and ***Reactive*** behavior. These behaviors are supported by CoSKS through the *Companion*.

3.6.1 Autonomous Behavior

An Autonomous System can be described as: *A system able to execute a number of basic steps without human intervention in order to achieve a given goal and to make decisions on its own, using high-level policies* [17,18].

An autonomous system/software infrastructure tailored to work on collaborative workspace shall be *human-centered*, i.e., executing its mission with a minimum support from users. Those systems ought to recognize people as intelligent agents or intervenient by operating and cooperating with them. Both human and system can (must) exchange information with each other, in order to enhance the whole collaboration process.

An autonomous system needs to be sure that its behavior and intervention near the users is coherent, robust, and adequate for the given situation thus, system needs to be aware of its environmental changes and react quickly to those changes.

Context-Awareness is, indeed, one of the key characteristics of an autonomous system since it needs information about the environment in order to keep the system in equilibrium. Systems with context-awareness need to be prepared to a number of potential pitfalls or rapid environmental changes, in order to maintain appropriate levels of predictability. With *Context-Awareness,* the system is capable to detect (quickly) ongoing operations performed by users and the consequences of those operations on environment.

Autonomous systems have to detect opportunities or anomalies where its intervention will be useful to the user. Those systems could have a *Proactive* or *Reactive Behavior*, as it will be explained in the following sections.

3.6.2 Proactive Behavior

Quoting Salovaara and Oulasvirta (2004) [19] *"Proactive systems adhere to two premises: 1) working on behalf of, or pro, the user, and 2) acting on their own initiative."*

The relevance of proactive systems within collaborative project teams must be analysed according to three major capabilities: (i) detect unforeseen problems before they become critical; (ii) provide some solutions for detected problems and; (iii) supply the user with understandable reports and alarms.

A common problem in the context of proactive behavior is the kind and impact of the interruption that occurs when a system performs a proactive action. Proactive systems must rely on appropriate interfaces to interact with users in a proactive way, not interfering with user's activity in an inconvenient manner.

Briefly, a proactive system is a system which: exceeds expectations, takes calculated risks, is an independent thinker, anticipates problems, seeks new solutions, questions/challenges and is multi-functional thinker [19].

In this sub-section it will be presented some situations[1] where a proactive behavior can be useful in a Collaborative Workspace, with the purpose of improving the management level of projects.

The following examples illustrate some situations where Companion proactive behaviour facilitates the daily work of the teams involved in the project:

- *When a Project Manager is creating a new Project or Task, Companion offers the possibility to see some reports about finished and relevant projects/ tasks*. These reports include similar projects/tasks, best practices, problems happened in similar projects/tasks, and solutions for those problems.

- *When creating a Project or Task, Companion proposes the best team to work on it.* This team is formed through the assessment of the actor's profile, the projects or tasks that he/she has previously performed.

- *The Project Manager must be constantly and automatically informed about the Project's/task's evolution.* Real-time information can prevent the spreading of existing problems throughout the project.

- *When watching a project/ task, Companion seeks, collects and organizes relevant information about the project/ task.* This relevant information can be an advice, document or warning that later on will be shown to the Project Manager or Actor.

- *Companion provides (*automatically and when it is appropriate*) important files or information that may help the Project Manager or Actor in a specific Project/Task.*

- *When a Project is ongoing, Companion checks whether the team involved in the project is appropriate for the project's needs*[2]*. If the team is unbalanced, Companion proposes*

[1] Notice that these situations are particularly related to the CoSpaces Project needs.

[2] These needs were described when the project was created, i.e., the number of participants needed to perform the project.

some actors to the Project Manager in order to balance the team. The choice of these actors is based on actor's profile and the number of similar projects that each actor is or was involved.

- *Companion is able to perform regular forecasts about Project or Task's evolution, to detect possible future problems or deviations.* If *Companion* detects future problems, it ought to inform the Project Manager or Actors.

- *After closing a Project or Task, the Companion systematizes all relevant information about it* (project or task) *and creates a report with that information.* The report will be sent to the Project Manager for future analysis.

- *During the meeting, Companion has context-awareness about the topics that are being discussed and automatically presents information or relevant files, in order to help and enhance the discussion of those topics.* This is one of the key benefits of a proactive software infrastructure. Its context-awareness makes possible the detection of what is happening and what kind of information the user needs to carry out his/her activity.

Example: During the meeting the users are discussing about roof leak related problems and at that time, *Companion* may provide documents or suggestions in order to help solving the problems.

- *Companion might also have context-awareness about the decisions made during the meeting.* It is possible to install cameras, vocal sensors, microphones, and light sensors to record all decisions[3].

- *When a meeting is completed Companion collects and organizes all the information and decisions made during the meeting or agendas.* After that, the component will create a minute that contains the information collected, organized and filtered.

- *When the project manager is creating a new meeting, Companion informs the project manager about the current state of previous issues*[4]. For each finished meeting the project manager gets a set of issues to solve. When he is creating a new meeting, *Companion* will inform the user about the state of those issues. This will help the Project Manager to be

[3] These decisions can be a set of issues to solve and they will be very important when finishing or creating a new meeting.

[4] These issues are the decision achieved in previous and finished meetings.

alert to both unsolved and solved issues. The possible states could be "**Planned[5], Ongoing, Completed and Critical[6]".**

- *When creating an issue Companion shows all finished issues and proposes a set of tasks to solve that issue.* These collection of tasks result through the assessment of previous solved issues.

- *When an Issue is completed Companion creates a report with all information and tasks that solved the issue.* This information will help the Project Manager to organize the tasks that will solve the issue - in Creation Issue phase.

3.6.3 Reactive Behavior

Reactive behaviour provides the capability to answer to the requests performed by users. Project participants usually have several requests targeting the solution a particular issue. Therefore, the CoSKS, using *Companion* capabilities, shall be prepared to quickly and appropriately respond to users' requests.

However *Companion* has to maintain its proactive behavior even when it is responding to an user request. For instance, if the user requests a particular report about an activity (just to see the progress or other information) and *Companion* detects relevant information about that activity, it will proactively provide the user with that information.

The following examples illustrate some situations where Companion reactive behaviour is useful:

- *Companion is able to provide Project Manager and Actors with appropriate knowledge about the project itself, task or issue's evolution (forecasts or current state). Companion* shall be prepared to respond promptly to all users' requests, with accurate information.

- *When a project manager is creating a new meeting, he/she can monitor the current state of pending issues. Issues labelled "critical" are (semi) automatically included into the decisional gate agenda by the Companion.*This function is very valuable because it

[5] Planned but has not been started, yet.

[6] Deadline reaching.

enhance the Project Manager's organization and makes the meeting's creation and planning easier.

- *Before creating a project, task or an issue the project manager can request the system to highlight some results, problems, and solutions about similar K-Elems that have been performed.* In the proactive behaviour, the system only shows this information after the creation of a K-Elem and the information shown is related to finished and similar K-Elems In reactive behaviour, the system will only display information about (finished or ongoing) requested items.

- *When a task is completed, the Companion displays a list of K-Elems used during the task execution, in order to help the user to select the most relevant K-Elem.* By selecting the most useful and relevant K-Elems, the actor will provide the system with more information about him/her. This information will increase the actor's ranking and the likelihood of be called in future projects.

Table 3.2 shows the applicability of proactive and reactive behavior within a project scenario and its different phases.

	CREATION	EXECUTION	COMPLETION
PROJECT	• *Important Inputs*[1]. • *Show Similar Projects*[2]. • Propose Actors[3] to handle the project.	• Check for Problems/Deviations. • *Propose and Provide Important Knowledge Elements*[4]. • Keep all actors informed about evolution of the project and provide some forecasts. • Inform all actors about the project's changes[5]. • Similar Projects. • Create Issue.	• Create a report[6] with all information[7] about the project. • Notify all actors about the completion of the project and offer to the project manager the report created.
MEETING	• Information[8] about previous tasks, related to previous meetings. • List of unsolved issues. • Create new issues. • *Create an agenda*[9]. • Create Free Topics[10].	• Propose important files about the topic/agenda that is being discussed. • Collect, organize and filter all the decisions that are being made.	• Create a minute about the meeting; • Show new issues, updated issues and solved issues.
TASK	• List of unsolved issues; • Create new issues. • Propose[3] the best Actors/Teams to handle the task. • Create Free Tasks[11]. • Show similar tasks.	• Check for Problems/Deviations. • Propose and Provide Important Knowledge Elements[4]. • Keep all actors informed of tasks evolution. • Inform all actors about the task's changes. • Show Similar Tasks.	• Create a report with all information about the task. • When all tasks – from an issue - are completed the Issue is closed. • *Show all used Knowledge Elements and update actor's raking*.
ISSUE	• Show information about similar and relevant issues.		• Create a report with all information and tasks that solved the issue.

Table 3.2 - Component's Behavior During Project's Life Cycle

Legend:

[1] Problems, Deviations and Best Practices.
[2] Reports about relevant and similar finished projects .
[3] Classified by actor's: availability, geography, Knowledge Elements etc.
[4] To enhanced problems solving.
[5] Such as updates, new dates, related problems, etc.
[6] These kind of report might contain graphics about the project's/task's evolution.
[7] Including errors found, solutions, project's evolution, involved actors, etc.
[8] Current state of each task: Ongoing, planned, completed and critical.
[9] Always related to an issue.
[10] Agendas not related to any issue.
[11] Tasks not related to any issue;

Italic and underlined words – Both proactive and reactive behavior.
Italic words – Reactive behavior.
Normal words – Proactive behavior.

3.7 The Interactions of Companion

In this section all the interactions among *Companion*, Mining Services, Knowledge Repository, Semantic Services, and the user will be illustrated and explained. Regarding these interactions and its conjugations, *Companion* may have different *Output Levels*. An *Output Level* is the level of information or help that *Companion* gives to the CoSKS user in order to help him/her in the management level of projects. The two existing Output Levels are:

- *Full Output level* – provides the user with documents, forecasts, best practices, detailed information about K-Elems, offers possible solutions for problems, proposes best actors to work on the K-Elem, check dates and deviations, collect and organize decision made during a meeting, create reports about completed K-Elems, find relevant information, and has context-awareness

- *Integrated Miner Output level* –offers less information and help then *Full Output level* but still offers a proper knowledge support level, such as, documents, provides the user with possible solutions for problems, check dates and deviations, create reports about completed K-Elems and find relevant information.

Companion has two possible configurations. One is the **Full Services Configuration**[7] (FSC), which is connected to external modules and the knowledge repository, providing the users with *Full Output level.* The other is the **Integrated Miner Services Configuration**, which is only connected to the knowledge repository and to its internal modules (IMSC). This configuration provides the users with *Integrated Miner Output level*.

In **Full Services Configuration** *Companion* is connected to the knowledge repository of the system, to the Mining Services and to the Semantic Services (both Mining and Semantic Services are represented by the orange boxes in figure 3.7).

[7] In this work, the *Companion* uses the **Full Services Configuration.**

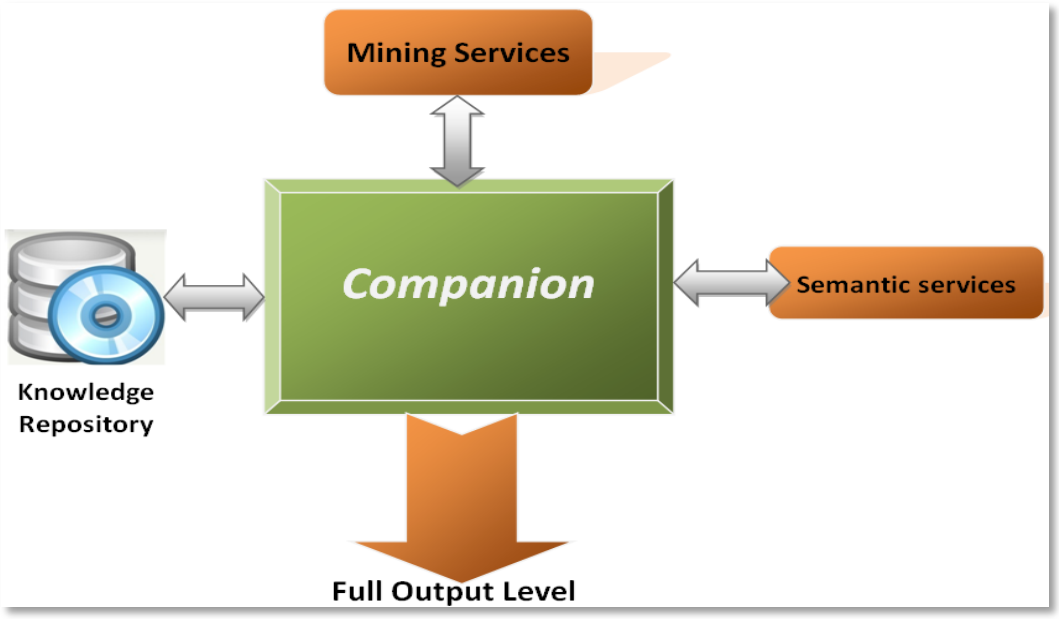


Figure 3.7 - Companion with Full Services Configuration

With this configuration, *Companion* can provide the user with *Full Output Level* (orange arrow in figure 3.7). This Full Output Level is achieved through the exchange of information between Mining Services and Semantic Services.

The use of Mining Services are very important because they offer a more flexible approach, which allows the application of different data exploration techniques to a large amount of data, in order to discover unknown relationships between variables or single data items. With an external and specialized module in Data Mining (Mining Services) and Semantic Services, *Companion* is able to deal with most accurate and reliable information to help the CoSKS user.

With the **Integrated Miner Services Configuration (IMSC)**, *Companion* is not connected to any external and specialized module in Data Mining but, it uses its own Data Mining services (gray box named *Integrated Miner* in figure 3.8). In this configuration, *Companion* is only linked to the knowledge repository of the system, in order to access and collect all the information existing in the system, but is not attached neither with external Mining Services nor external Semantic Services (figure 3.8).

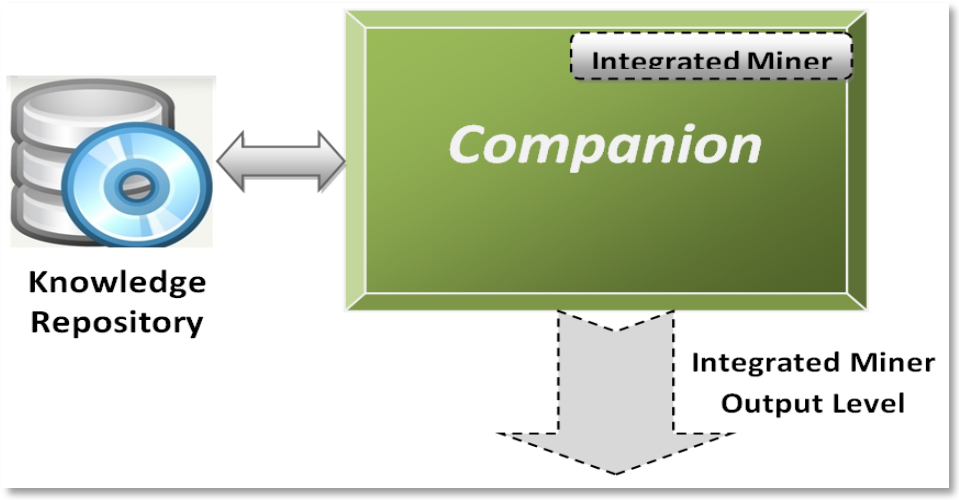


Figure 3.8 - Companion with Integrated Miner Services Configuration

Once the *Companion* does not have any external Mining Services, it only uses its own integrated miner services, thus the output level offered to the CoSKS user is less powerful than using external mining services.

These integrated miner services can use a folder, a database or an external server, providing mining services with reports or information about previous projects and user activities. The *Companion* can use this folder or database to extract and assess data information.

It is important to refer that the integrated miner services must use the eXtensible Markup Language (XML) or document that stores all the semantic definitions used in the system which *Companion* is connected. These semantic definitions replace the use of external Semantic Services and they are crucial to guarantee a perfect comprehension of the *Companion* about the concepts and names used in system.

This structure is very useful since using its own integrated miner services module, *Companion* proves that is able to be connected and used in any existing system (including legacy systems) providing those systems with a valuable knowledge support level.

Since this work intends to create, design and implement a component to be easily plugged into any existing system, the two configurations presented here (FSC and IMSC) are very useful. The IMSC offers a lower level of knowledge support then FSC**,** but it makes the *Companion* an easy pluggable component while, the FSC must be used in a system previously built in concerning the *Companion* needs.

3.8 The Companion Behavioral Services

Within the CoSKS framework, the Companion uses two classes of services, namely Mining Services and Semantic Services (figure 3.9). Whilst the former handle reasoning-related requirements, the later tackles the ontology-related needs.

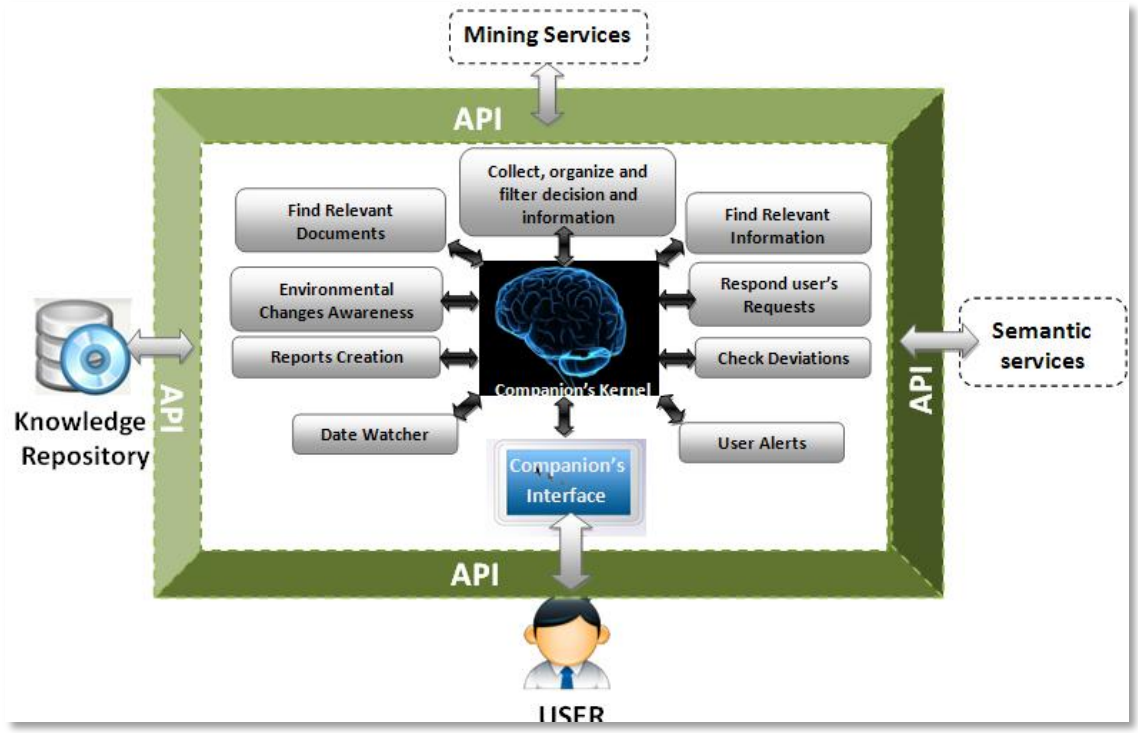


Figure 3.9 - The Companion services

The Companion architecture (figure 3.9) is organized around the *Companion* Kernel supported by nine services, namely: 1) Date Watcher; 2) User Alerts; 3) Reports Creation; 4) Check Deviations; 5) Environmental Changes Awareness; 6) Respond User's Requests; 7) Find Relevant Documents; 8) Find Relevant Information; and 9) Collect, organize, and filter decisions/information.

The service **Respond User's Requests** represents the *Companion*'s Reactive behavior and all the other services represent the *Companion*'s Proactive behavior.

Companion Kernel

It acts like a maestro of *Companion* internal services. It orchestrates the execution of services to provide the appropriate answer to a given user request, which may involve not only the *Companion* internal services but also both mining and semantic services.

Date Watcher

This service is responsible for checking all the deadlines existing in the projects, tasks, issues or meetings. For each Knowledge element there are several deadlines to be respected and fulfilled. This service keeps the user informed about all deadlines, informing him/her about critical or reached deadlines.

User can program the interval of time to be informed and the regularity which Date Watcher service will check the dates.

User Alerts

Companion uses this service to provide the proper alert to the user when it is required to offer him important information, documents, and forecasts it will employ this service to provide the proper alert to the user. User Alerts service is notified by *Companion* Kernel about which type of alert is needed. User Alert processes that notification, creates an alert with text and an appropriate symbol, for the alert requirement, and sends it to *Companion* Kernel.
After this process the *Companion* Kernel sends the alert to *Companion* Interface in order to be shown to the target user.

Reports Creation

This service creates all the reports and is also responsible for creating the Meeting Minutes.

When closing a knowledge element it is important to create a report that summarizes the whole process. This report is crucial to help the user when he/she is creating a project, task,

issue or meeting, because it gives information about similar and relevant finished Knowledge elements.

These reports can also be requested by the user during knowledge element execution, to aid the user to make important decisions.

All the created reports are stored in BSCW[8] and knowledge repository.

Check Deviations

Companion uses this service to check the project or tasks deadlines and to check whether the project or tasks requirements will or are being fulfilled. The Decisional Gates concept is especially important because it provides important milestones to be respected and information about current project state.

This service sees both projects and tasks as a sequence of stages, through Decisional Gates information. Check Deviations works together with Mining Services in order to forecast possible and future problems/deviations or existing problems/deviations.

Environment Changes Awareness

This is one of the most important services in the *Companion* architecture. With this service, *Companion* is enriched with *Context-Awareness* skills which make it able to detect environmental changes.

Companion can adapt its behavior and predictions to the new environmental changes. This Environmental Changes Awareness service is particularly used during a meeting execution.

During a meeting, several agendas are discussed and with this service *Companion* can detect which topic is being discussed. When receiving this information, *Companion* requests the service 7 (**Find Relevant Documents)** to find all the important documents that could help the agenda in discussion.

[8] Please see section 3.2 on page 23.

This service also detects when projects, tasks, issues or meetings are created, closed or updated.

Respond User's Requests *(Reactive Behavior)*

This service is the only reactive service from *Companion*, every time a user needs a report, information, forecast or a document he/she uses the *Companion* interface to send his/her request to this service. In fact this service is indirectly associated with all *Companion* services, i.e., in order to supply the user needs, this service will send the user request to *Companion* Kernel, which will process that request and send it to any other existing service in order to satisfy the user.

Find Relevant Documents

With this service, *Companion* provides the users with historical information about previous and similar projects, tasks, issues, and meetings.

Within a project scenario the capability of seeing reports about finished K-Elems could be vital to the success of the collaboration. By learning from previous mistakes and problems, the Project Manager can anticipate and avoid possible repetitions.

This service is an important ally of **Environmental Changes Awareness** service.

Find Relevant Information

When using the system which *Companion* is connected, the user shall be informed about all changes occurred in projects, tasks, issues, meetings and profiles. This service responsibility is to seek and find those changes and select each one are relevant and important for the logged user. This service sees each user and situation as a particular case.

For instance, when a project Manager is creating a Project this service (in collaboration with Mining Services) assesses that particular project and its needs. After that it seeks for relevant actors to work on that project, important documents that can be useful, particular aspect to be taken into account with the project and sends that information to the user.

Collect, organize and filter decisions and information

The information provided by user must be collected, filtered and organized before its usage.

For instance, during the meeting execution important decisions and conclusions are made and achieved. This service collects filters and organizes that information in order to help the Minutes creation. When a meeting is finished, this service sends the meeting information to the **Reports Creation** service, which will create a minute about finished meeting.

The Companion Interface

Companion also has its own interface to avoid conflicts between system's interface (system which *Companion* is connected) and the interface where *Companion* shows the information to the user. *Companion*'s interface is, indeed, the bridge between *Companion* services and the user.

It is important to refer that in the CoSpaces project and due to the project requirements, the *Companion* interface was embedded into the CoSpaces interface.

API – Application Programming Interface

Companion accesses other services (Mining services, Semantic Services, and Knowledge Repository) trough The *Companion* Application Programming Interface (API) and vice-versa.

Chapter 4 – Implementation

In this chapter the adopted technologies and the whole implementation process are described. Additionally, it presents a quick reference manual that illustrates the main functionalities of the *Companion* in CoSKS and shows the functional and structural view of the *Companion* Software Infrastructure.

4.1 Technologies Adopted In This Work

The conceptual work previously presented could be implemented using many technologies. Table 4.1 shows the main technologies used to implement this work.

Technology	Description	Why was it used?
Netbeans	It is both a platform framework for Java desktop applications, and an IDE for developing with Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala and Clojure [21].	The NetBeans enables the developers to rapidly create web, enterprise, desktop, and mobile applications using the Java platform, as well as JavaFX, PHP, JavaScript and Ajax, Ruby and Ruby on Rails, Groovy and Grails, and C/C++ [21].
Hibernate	It is an ORM[9] library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database [22].	Its primary feature is to map Java classes to database tables (and from Java data types to SQL[10] data types). It also provides data query and retrieval facilities [22].

[9] Object-relational mapping.

[10] Structured Query Language.

Struts 2	It is an open-source web application framework for developing Java EE web applications. It uses and extends the Java Servlet API to encourage developers to adopt a Model-View-Controller[11] (MVC) architecture [23].	It separates the *model* from the *view* and the *controller*. It also provides the controller (a servlet known as `ActionServlet`) and facilitates the writing of templates for the view or presentation layer in JSP [23].
MySQL	It is a relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases [24].	It is an Open source database application, fast performance, high reliability, ease to use and costs saving [24].
Java Server Pages	It is a server side Java technology that allows software developers to create dynamically generated web pages, with HTML, XML, or other document types, in response to a Web client request to a Java Web Application container (server) [25].	It allows Java code and certain pre-defined actions to be embedded into static page content and compiled on the server at runtime of each page request. It is also a faster/easier way to display parameter values [25].
Tomcat	It is an open source servlet container. Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications and provides a "pure Java" HTTP web server environment for Java code to run [26].	Implements the Servlet 2.4 and JSP 2.0 specifications. Reduces garbage collection, improves performance and scalability. Native Windows and Unix wrappers for platform integration and faster JSP parsing [26].
Visual Paradigm for UML	It is a UML CASE Tool supporting UML 2.1 and the Business Process Modeling Notation (BPMN). It provides business process modeling, an object-relational mapping generator for Java, .NET and PHP [27].	Supports 13 UML diagram types: Class diagram; Use case diagram; Sequence diagram; Communication diagram State; Activity diagram; Component diagram; Deployment diagram; Package diagram; Object diagram; Composite structure diagram; Timing diagram Interaction and Overview diagram [27].

Table 4.1 –Used Technologies

[11] ***Model*****:** application logic that interacts with a database; ***View* :** HTML pages presented to the client; **C*ontroller**:* instance that flows information between view and model.

4.2 The *Companion Software Infrastructure*

In order to design and implement the *Companion* Software Infrastructure, both Unified Modeling Language (UML) and ICE[12] methodology were used.

The UML is a standard general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system. It is, indeed, a set of graphical modeling notations (shapes to construct diagrams) and textual modeling notations (syntax that tells how the shapes can be combined).

It is used to understand, design, maintain, and control information about a software system. UML helps to visualize, and document models of systems or processes, including their structure and design, in a way that meets the requirements specifications. It also helps stakeholders to understand what the system will be and what the possible available options are.

In this work UML was used in order to support the entire project lifecycle and to create graphical representations of the *Companion* Software Infrastructure.

The ICE methodology is used to guide the development of computer systems. Essentially, the ICE divides a system into three layers, namely:

- ***Interface:*** It contains all the classes that manage the system's interactions with the outside world.
- ***Control:*** Acts as a bridge between Interface and the Entity layer, receiving and processing requests from the Interface, using the data managed by the Entity layer.
- ***Entity:*** It contains all the classes that manage the data used by the system.

The *Companion* Software Infrastructure will be presented in two views: *Functional View* (using UML Use Case Diagrams) and *Structural View* (using Class Diagrams).

4.2.1 Companion Functional View

Companion Functional View is expressed using UML Case Diagrams showing the main functionalities offered to participants and Project Manager.

[12] Interface, Control, and Entity.

Both Participants and Project Manager have different access rights and roles within a project scenario.

On the one hand the Project Manager can create a Knowledge Element, observe and participate in its execution and perform its closure. On the other hand, Participants can only observe and participate in the project, task or meeting execution.

For the Project Manager point of view, a Knowledge Element is something to be monitored from the beginning until the end. He is responsible for creating a Knowledge Element, for making decisions during its execution, for choosing actors to associate with the Knowledge Element, and for its success.

Participants can only work in Knowledge Elements and follow the orders from the Project Manager, i.e., they are not able to create or close a Knowledge Element.

Use Case Diagrams for Project Manager

- ***Project Scenario***

Figure 4.1 shows the interactions between Project Manager and the Companion in a project scenario.

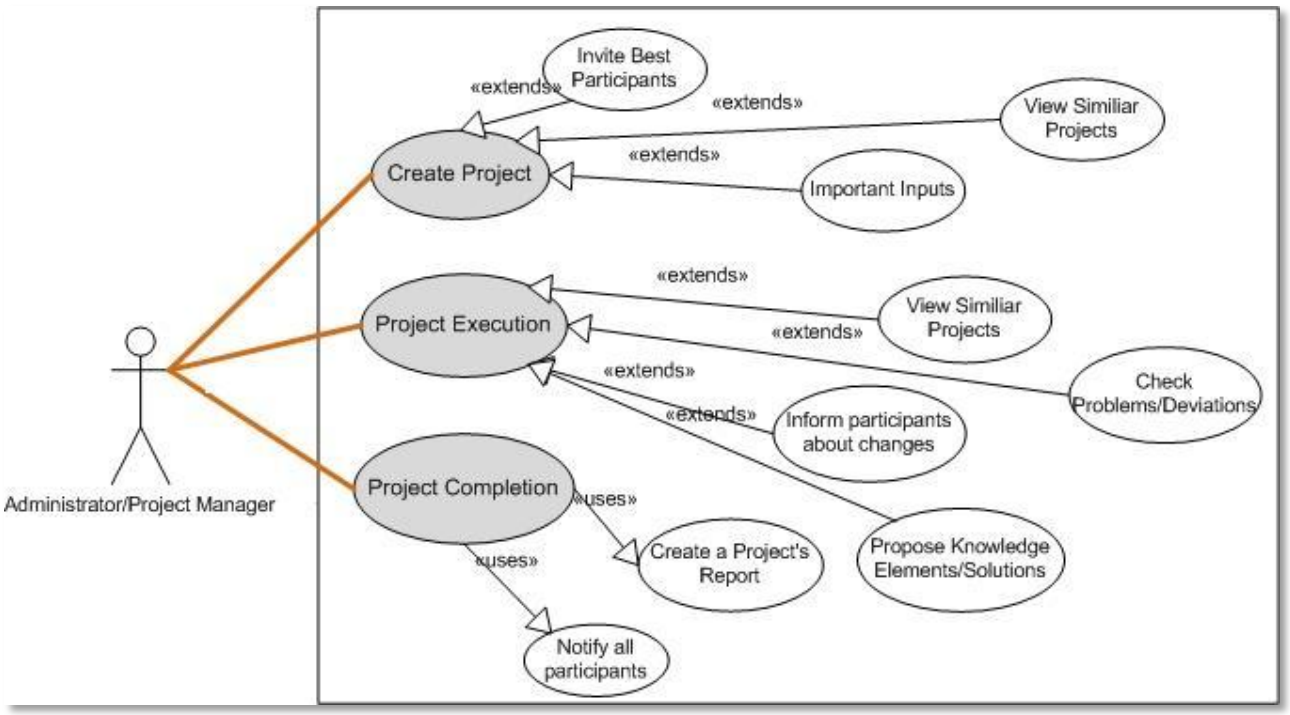


Figure 4.1 - Use Cases Project Scenario (Project Manager)

When creating a project, the *Companion* automatically provides the Project Manager with some reports about similar and relevant finished projects, in order to help him to avoid some common errors. *Companion* also informs Project Manager about important inputs to take into account, such as: best actors' profiles to work on the project, important documents to read before starting the project, and some forecasts about possible deviations or issues likely to be found, and some feasible solutions.

During the project execution, Project Manager is able to see some reports about relevant and similar finished projects. Both reports and documents can periodically and automatically be offered to the user (*Proactive Behaviour)* or they can be requested by the user whenever needed (*Reactive Behaviour)*.

Companion also checks for possible problems or deviations and notifies all actors working on the project, in order to reduce decision making process to solve or help solving the problem.

The Project Manager is able to upload the system with useful and new Knowledge Elements. Once the project is update, the actors involved in the project are informed.

When a project is completed, the Project Manager is informed and he has to close the project. When the Project Manager closes the project, *Companion* creates a report which summarizes all project main features and activities. If the deadline of the project is reached, the Project Manager can extend the end date of the project.

- ***Meeting Scenario***

The figure 4.2 shows the interactions between Project Manager and the Companion in a meeting scenario. Similar to the Project scenario, the Project Manager is responsible for creating, observing, participating, and closing a meeting.

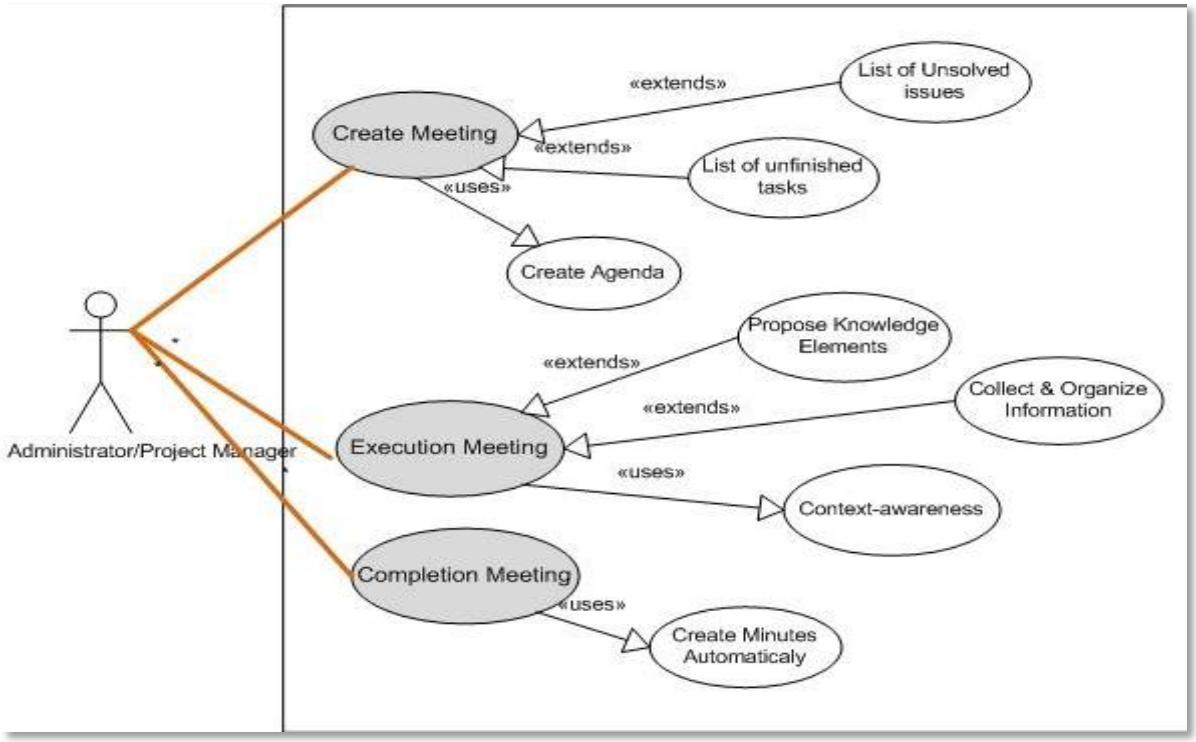


Figure 4.2 - Use Cases Meeting Scenario (Project Manager)

When starting a meeting the Project Manager can create an agenda for all unsolved issues or create a free topic to discuss the progress of ongoing tasks. *Companion* provides the Project Manager with the list of all unsolved issues and unclosed tasks.

During the meeting execution the *Companion* may recognize and know what information is important to help the discussions (*Context-awareness* concept). By knowing that, *Companion* can automatically offer documents or information in order to help the discussion and the decision-making process.

When an agenda ends, the *Companion* makes a few and short questions[13] to the Project Manager in order to understand which decisions were made. *Companion* will collect and filter that information to automatically create a minute of the meeting, when the meeting is closed.

When the meeting finishes, the Project Manager closes it and the *Companion* offers him/her two possibilities, namely:

[13] Notice that these questions can be skipped.

- ***Create automatically a Minute with the collected information –*** With this option the *Companion* creates automatically a minute with the decision made during the meeting. By choosing this option the Project Manager does not have to insert any additional information.

- ***Create a manual Minute*** – By choosing this option the Project Manager has to create the minute without any help from *Companion*.

- ***Task & Issue Scenario***

Since an Issue is, indeed, a set of linked Tasks, both Tasks and Issues Use Cases Diagrams are presented in just one Use Case Diagram (figure 4.3).

When a Project Manager is creating an issue**,** *Companion* provides him with two types of support. One is the possibility to see some reports about relevant and similar solved issues, in order to be aware of frequent problems and the time needed to solve them. The other is, based on previous issues and their solutions, *Companion* offers some sequences of tasks that can solve the created issue. For each sequence of tasks, *Companion* informs the Project Manager about their rate success.

When creating a task, the Project Manager is able to see useful reports about similar and finished tasks, a list of unsolved issues (to associate the task to one of them), and a list of possible participants[14] to work on the task.

Figure 4.3 shows all the interactions between Project Manager and *Companion*, to both task and issue scenario.

[14] Based on their profiles and performed tasks.

.

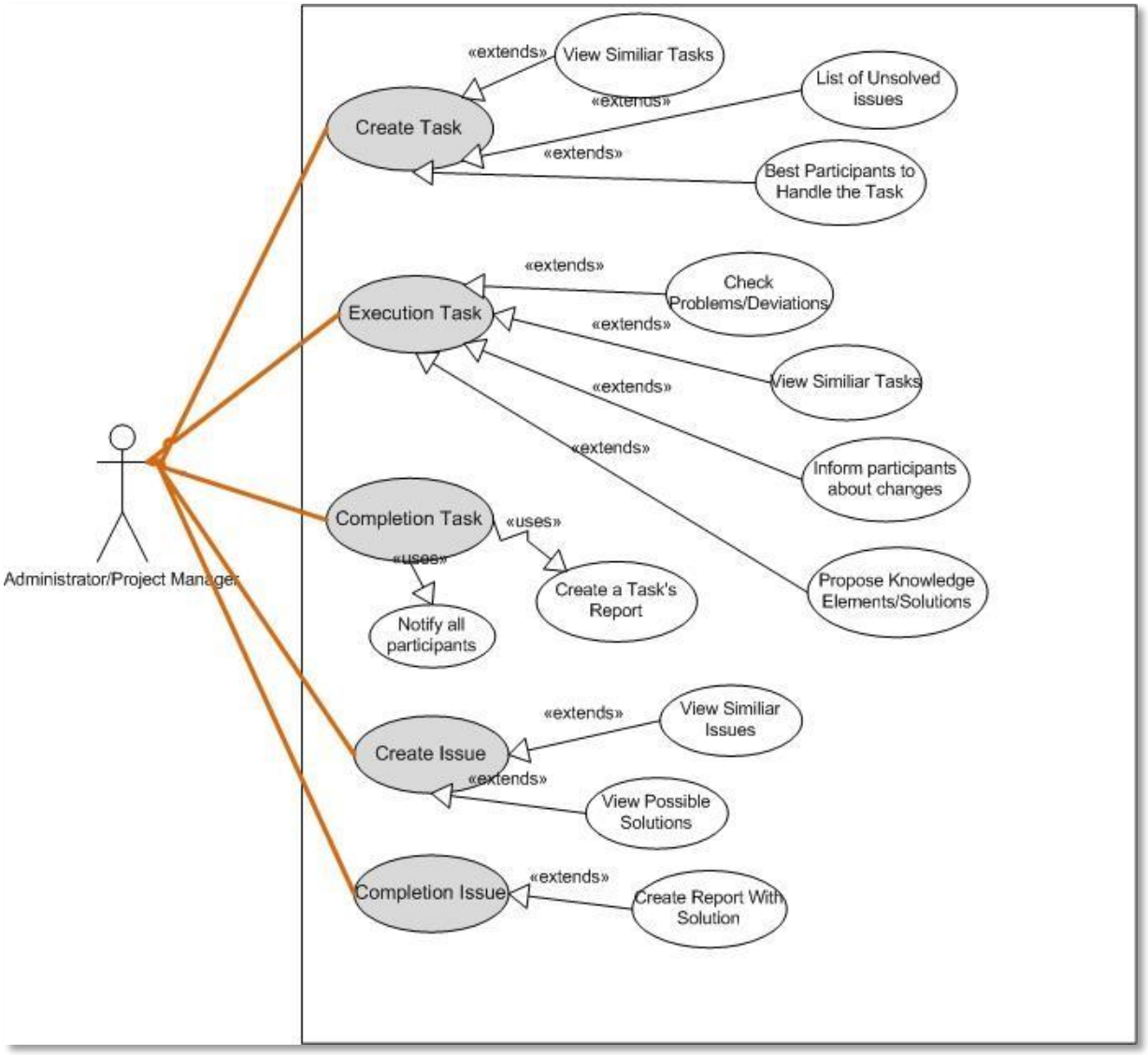


Figure 4.3–Tasks & Issues Interactions Use Cases (Project Manager)

During the execution of a task, the Project Manager can read some reports about relevant and similar finished tasks and download important documents, in order to help the task execution and its success. Similar to the Project scenario**,** both reports and documents can periodically and automatically be offered to the user (*Proactive Behaviour)* or they can be requested by the user whenever needed (*Reactive Behaviour).*

Project Manager and Participants are able to upload the system with useful Knowledge Elements.

Companion checks (proactively or reactively) for possible problems or deviations and notifies all actors working on the task.

When a task is completed, *Companion* informs the Project Manager about that and the task has to be closed. Once the task is closed, *Companion* automatically creates a report about the main activities, problems, deviations, and documents read during the execution of the task. If the deadline of the task is reached, the Project Manager can extend the end date of the task.

Once all tasks from an issue are closed, *Companion* notifies the Project Manager to close that issue and creates a report with the sequence of tasks that solved the issue, the main problems, and activities that happened during its execution.

Use Cases Diagrams for Participants *(Ordinary Users)*

- ***Project Scenario***

During the Project Execution both Project Manager and Participants involved in the project receive the same information[15] from *Companion* (figure 4.4).

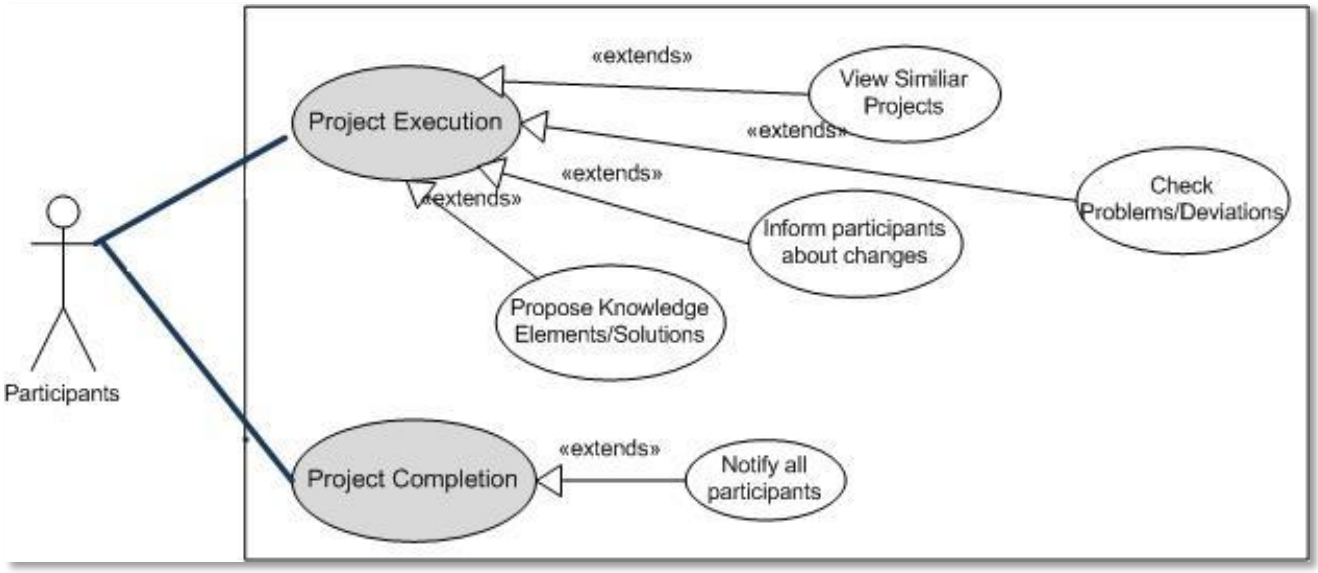


Figure 4.4 - Project Interactions Use Cases (Participants)

[15] Please see the explanation in page 64.

When a project is completed or has reached its deadline, the participants only see a warning informing that the project finishes, and when the Project Manager closes the project they are informed.

- ***Meeting Scenario***

In a meeting scenario, the participants can only observe and participate in the meeting execution (figure 4.5). The interaction of the participants during the meeting execution is similar to the interaction of Project Manager described on page 66.

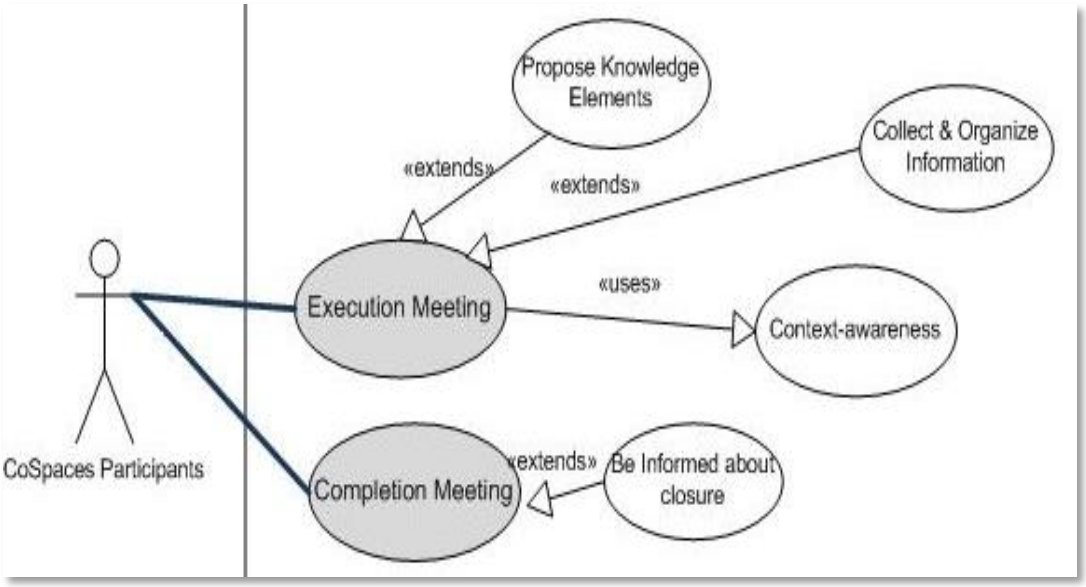


Figure 4.5 - Meeting Interactions Use Cases (Participants)

When a meeting finishes, the participants see a warning informing that the meeting is over, and when the Project Manager closes the meeting they are informed.

- ***Task & Issue Scenario***

When executing a task both Project Manager and Participants get the same information[16] from *Companion* (figure 4.6).

Once a task or an issue finishes, the participants receive a warning notifying that the task or issue is over, and when the Project Manager closes the task or issue they are informed.

[16] Please see page 67.

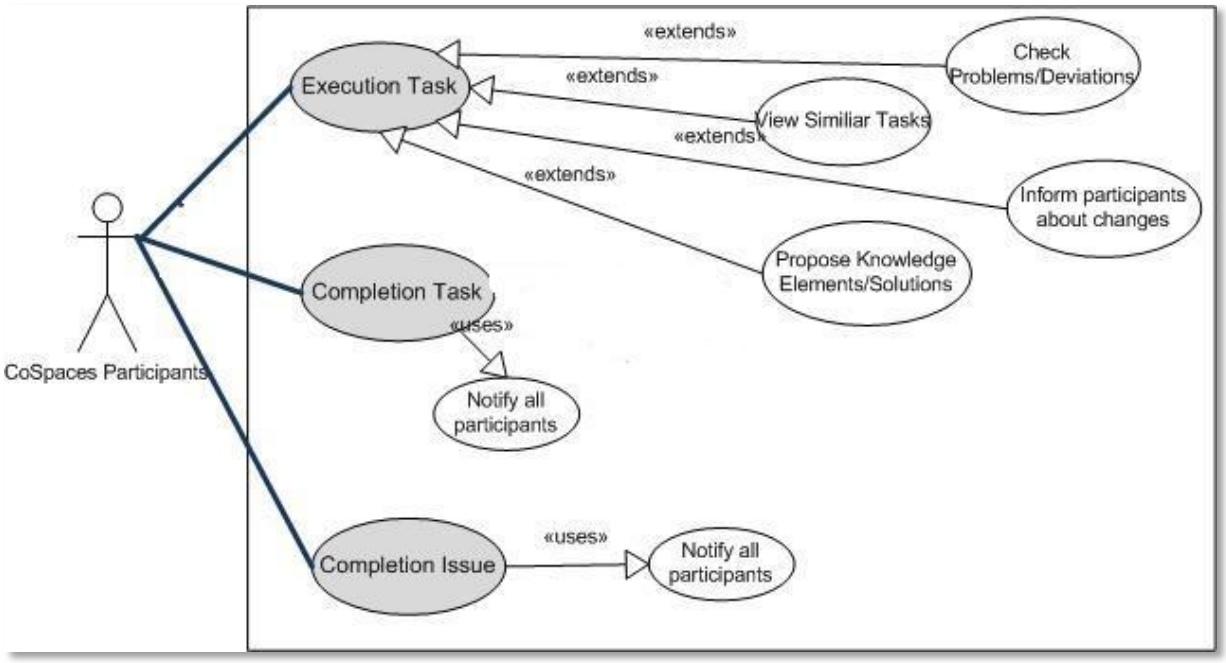


Figure 4.6 - Tasks Interactions Use Cases (Participants)

4.2.2 Companion Structural View

Figure 4.7 shows the three layers (**Interface Layer, Control Layer,** and **Entity Layer**) of the ICE architecture representing the *Companion* main classes.

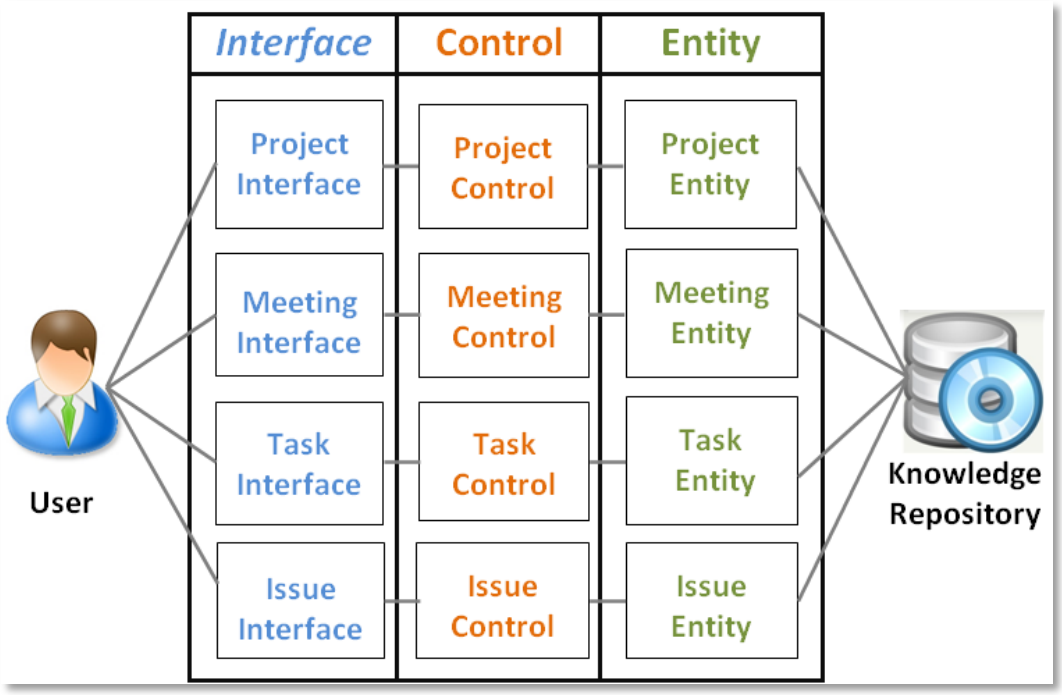


Figure 4.7 – ICE Class Diagram

These classes contain several attributes and functions, with their parameters. For the sake of clarity, each class is only represented with one attribute and one function with its parameters. The appendix A on page 89 shows a complete view of the classes.

4.2.2.1 Interface Layer

This layer contains all the classes that manage the system[17] interactions with the users, i.e. this class controls the exchange of information involving, *Companion*, CoSKS portal, and users.

There are four main classes to handle that information, namely Project_Interface, Meetings_Interface, Tasks_Interface,and Issues_Interface.

Projects_Interface

This class manages the information sent and received to/from user to create, execute and complete a project (figure 4.8).

These functions allow the user to receive information from *Companion*, send requests to *Companion* or to the CoSKS or read reports about the project.

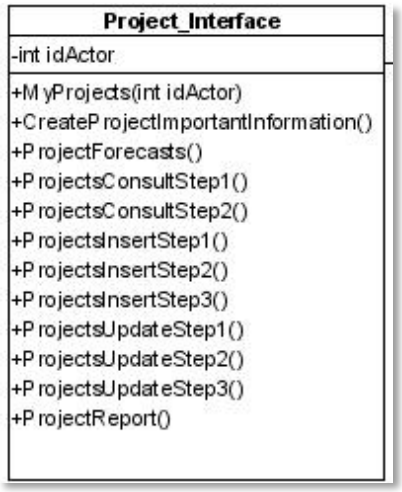


Figure 4.8 - Project_Interface

[17] Which *Companion* is connected.

Meetings Interface

This class manages the information sent and received to/from user to create, execute and complete a meeting (figure 4.9). With these functions the user can obtain important information and manage the meeting.

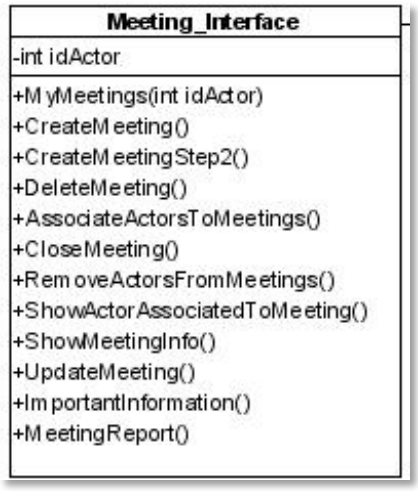


Figure 4.9- Meetings Interface

Tasks Interface

Similar to the Project Interface class, this class manages the information sent and received to/from user to create, execute, and associate actors with the task. It also provides the Project Manager with reports and documents, forecasts, and possible deviations. (figure 4.10).

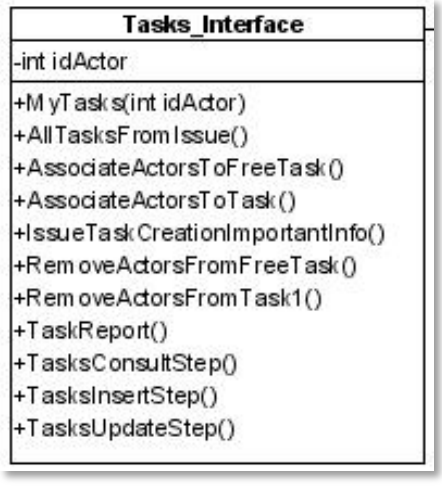


Figure 4.10- Tasks Interface

Issues Interface

This class enables the exchange of information involving, CoSKS portal, *Companion* and the CoSKS user (Figure 4.11), i.e., it allows the management of issues (creation, associate tasks, see similar and finished issues, etc.).

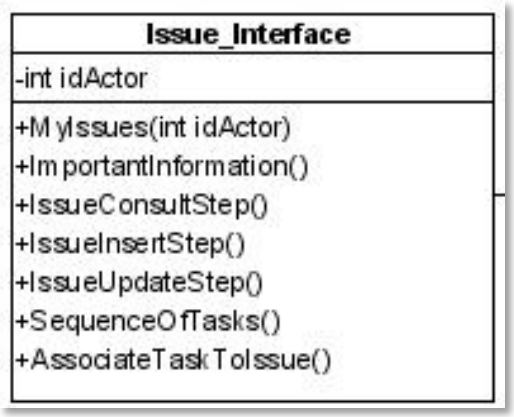


Figure 4.11 - Issue Interface

4.2.2.2 Control Layer

These classes are the bridge between Interface and the Entity layer, enabling the transactions between both of them. These classes send/receive information to/from the Interface classes (JSP) and send/receive that information to/from Entity Classes (classes that stores and gets information from the Knowledge Repository).

There are four main classes to handle the exchange of information, namely *Project_Control* (figure 4.12), *Meetings_ Control* (figure 4.13), *Tasks_ Control* (figure 4.14), and *Issues_ Control* (figure 4.15).

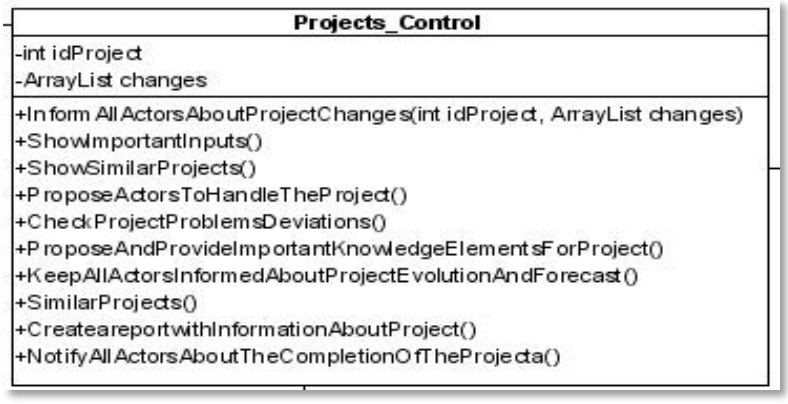


Figure 4.12 - Project Control

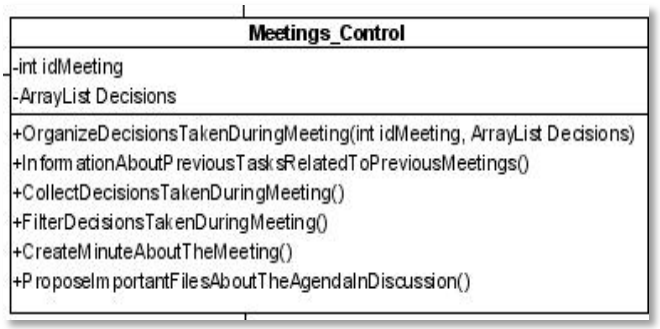


Figure 4.13 – Meeting Control

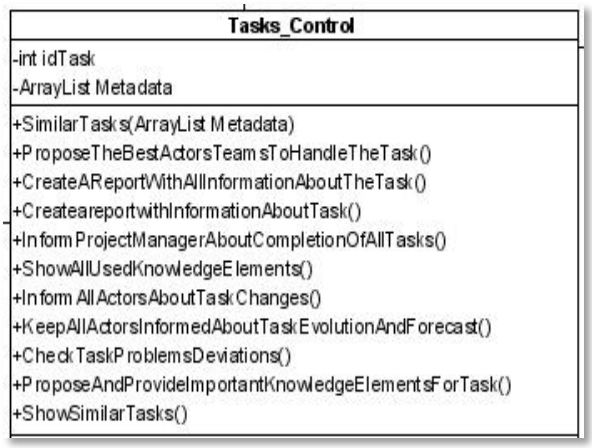


Figure 4.14 – Task Control

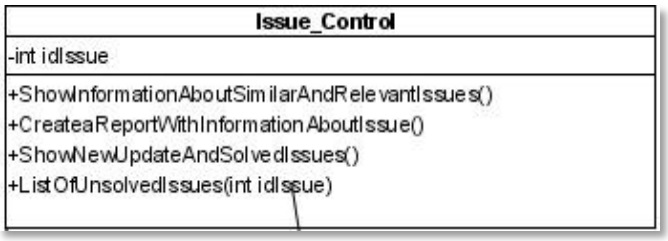


Figure 4.15- Issue Control

Those classes have the functions to send or receive information about projects, meetings, tasks, and issues (to or from the Interface classes), compute that information and forward it to the Project, Meeting, Task or Issue Entity class, respectively (in order to store or get information from knowledge repository).

4.2.2.3 Entity Layer

As previously stated, the Companion functionalities were implemented within the CoSKS portal. The Knowledge Repository used in CoSKS portal was implemented according to the Entity-Relationship Diagram (ERD) showed in Appendix B on page 93. *Companion* uses that Knowledge Repository in order to perform its functionalities.

The Entity Classes (Project_Entity, Meeting_Entity, Task_Entity, and Issues_Entity) manage the access to the Knowledge Repository.

Projects Entity

The functions in this class manage the information existing in the tables (of the Knowledge Repository) that have the information about projects (figure 4.16).

Figure 4.16- Projects Entity

The Figure 4.17 shows three different tables (Projects, ProjectsActors, and ActorsType), where information about projects is stored. *Companion* uses those tables in order to extract information about projects.

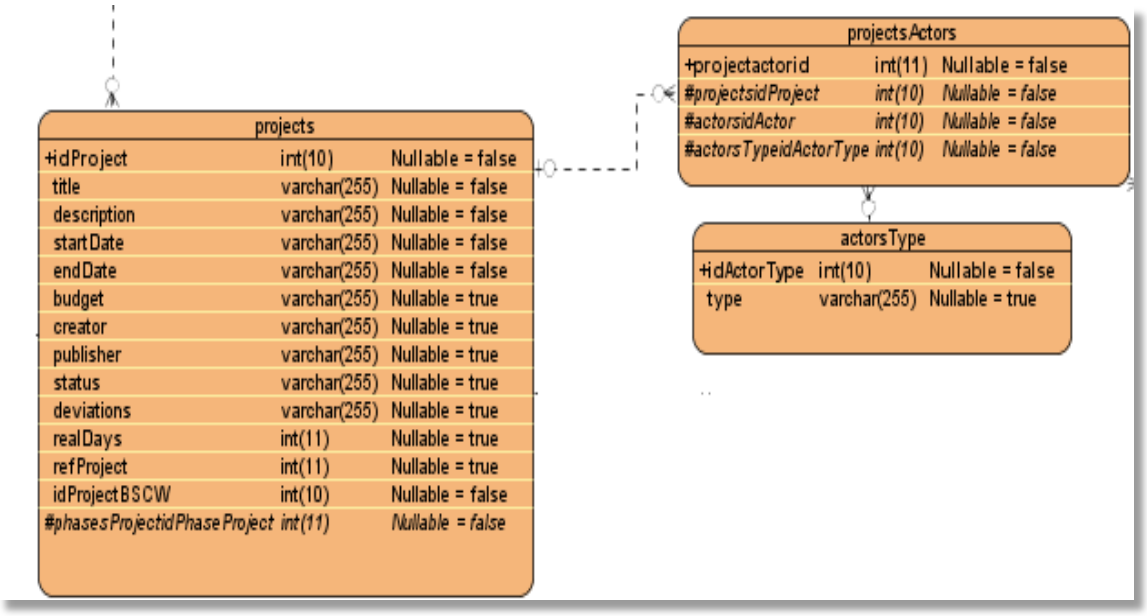


Figure 4.17 – Excerpt of the Companion DER (Project Connections)

The *Companion* uses the information computed by the Data Mining services (through the table **Projects)** to get forecasts about possible problems or deviations, to propose relevant documents to help the execution of the project, and to create a final report about project.

ProjectActors and ***ActorType*** tables get which actors are working on the project and what their roles on that project are. *Companion* uses this information to recommend actors in future and similar projects.

Meetings Entity

The functions in this class control the information existing in the tables (of the Knowledge Repository) that have the information about meetings (figure 4.18).

Figure 4.18 - Meetings Entity

The figure 4.19 shows the tables used in the creation of a Meeting. The table **Agendas Object** (Meeting Input) and the table **Minutes Object** (Meeting Output) support the definition of a Decisional Gate[18].

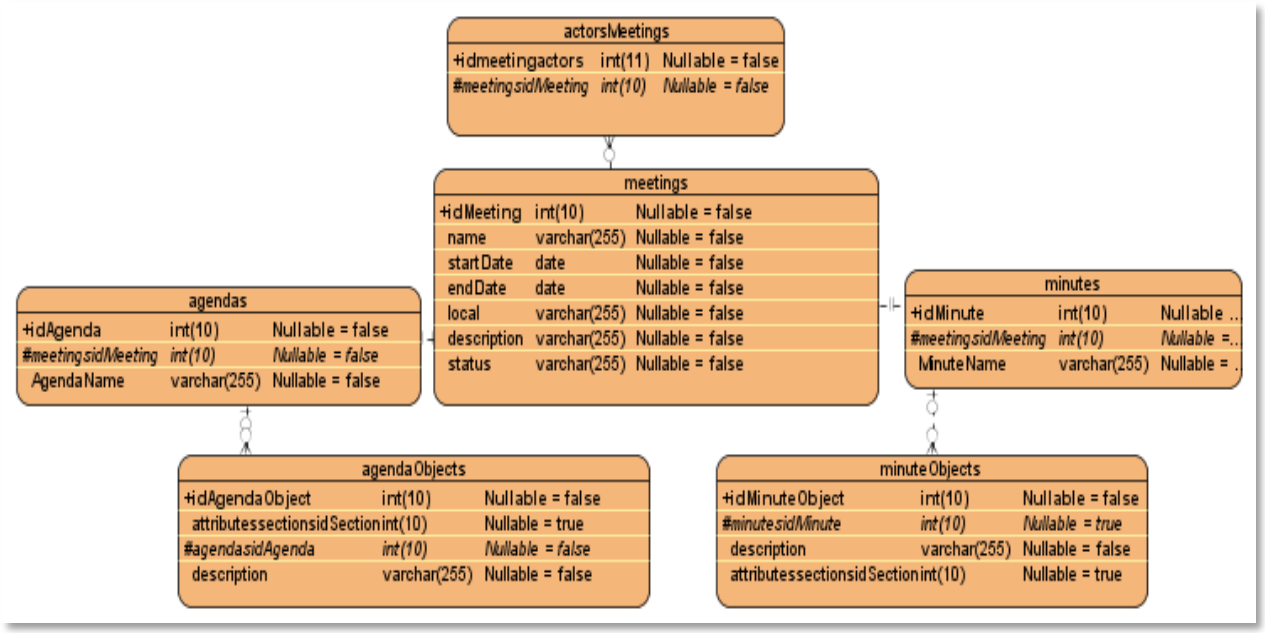


Figure 4.19 - Excerpt of the Companion DER (Meeting Connections)

Meetings table has the profile and information about all created meetings. **Actors Meeting** table has all the actors involved in the meeting.

[18] Please see table 3.1 on page 19

The **Agendas** table contains the list of **Agendas Objects**[19]**.** With this information *Companion* knows which topic is being discussed and according to that topic, *Companion* offers important documents to support the discussion and collects all the decisions made.

Minutes table is the table where *Companion* stores the reports with the information about the meeting and the decisions made. *Companion* also creates a **Minute Object** (solutions achieved during meeting and future problems to solve) for each **Agenda Object**.

Tasks Entity

The functions in this class control the information existing in the tables (of the Knowledge Repository) that have the information about tasks (figure 4.20).

Figure 4.20 - Tasks Entity

As Figure 4.21 shows, the table Tasks is connected to the Issue table (an issue is a set of linked tasks), to the table project (each task is related to a project), and to the Task Documents and Task Actors table.

[19] Topics that will be discussed during the meeting.

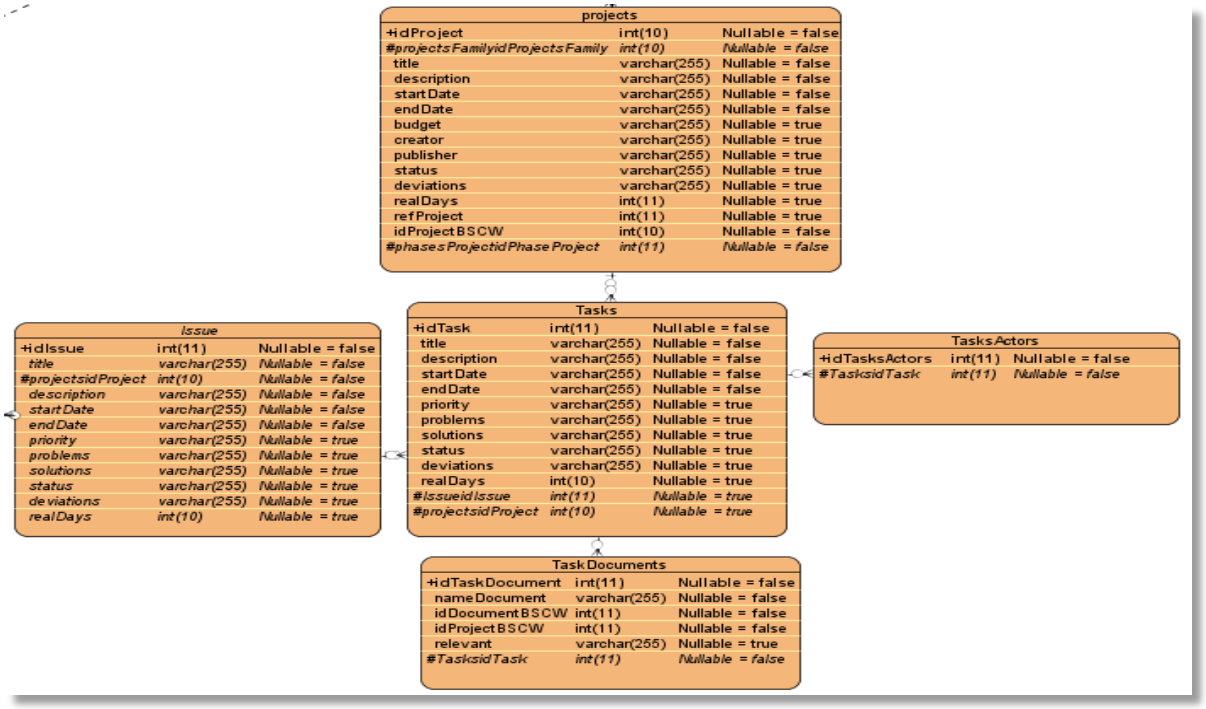


Figure 4.21 - Excerpt of the Companion DER (Tasks Connections)

The *Companion* uses the information computed by the Data Mining services (through the table **Tasks)** in order to get to compare tasks, to find relevant documents to support the task, to get forecasts about possible problems or deviations, and to create the final report about the task.

Task Actors table has the information about the actors working on a task. By using that information the *Companion* is able to suggest some actors to deal with future and similar tasks.

Task Documents stores all the important documents accessed or uploaded by an actor during the task execution.

Issues Entity

The functions in this class deal with the information existing in the tables (of the Knowledge Repository) that have the information about issues (figure 4.22).

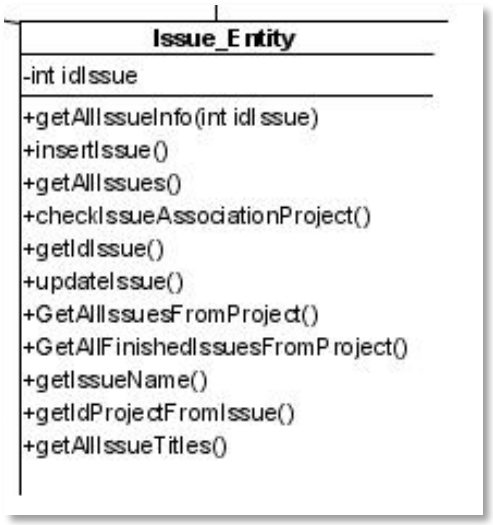


Figure 4.22 - Issue Entity

The Figure 4.23 presents the tables connected to the Issue table, namely: Project, IssuesDocuments, IssuesAgendas, and Tasks.

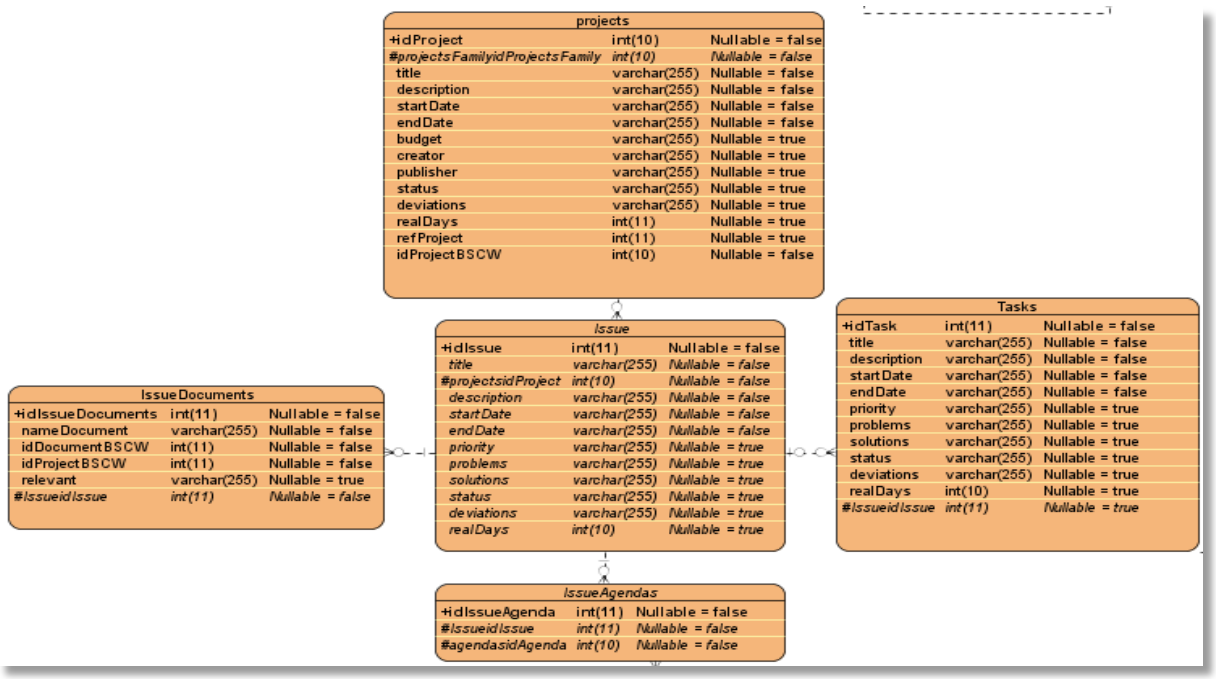


Figure 4.23 - Excerpt of the Companion DER (Issue Connections)

Companion uses the Data Mining services to compute essential information about an issue. This information allows it to assess several key points in order to show similar and finished issues, to find relevant documents to help the issue execution, to get forecasts about

possible problems, and to create the final report about the issue. With the information provided by the Data Mining services, *Companion* can know which tasks are associated with an issue (table **Issues** and **Tasks** are connected with a relation 1-n), and show to the Project Manager which tasks are relevant and important to solve a specific Issue.

The table **Issue Agenda** associate an issue to an agenda, in order to be discussed during a meeting.

Issues Documents stores all the important documents accessed or uploaded by an actor during the issue execution.

The Figure 4.24 shows the tables which stores the information about actors and their roles, within a project scenario.

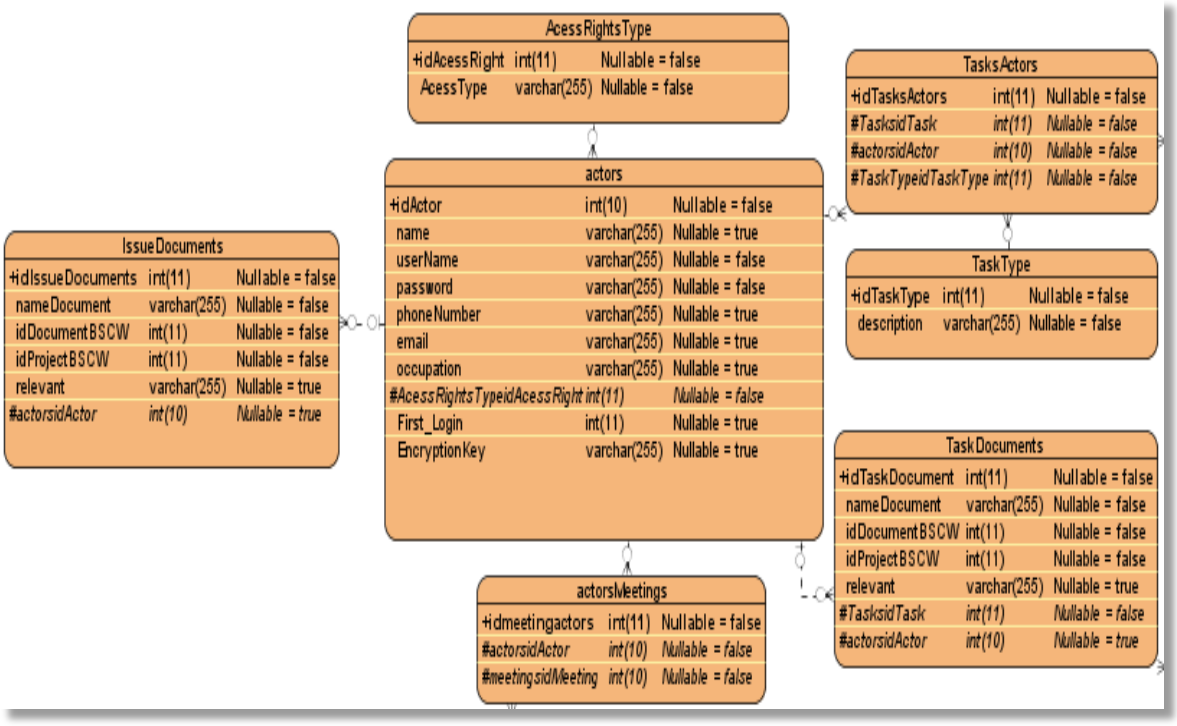


Figure 4.24 - Actor profile and his/her interactions

The **Actors** table stores the profile and information of all actors. With this information *Companion* can assess the best actor to work on a project or task. The table **Access Rights Type** stores the access rights of the actors in the project. By knowing the access rights of an actor, *Companion* is able to give different information to each actor.

4.3 User Reference Manual

In order to illustrate the operation of the *Companion* into CoSpaces Project, it will be presented a quick reference manual that illustrates some of the main functionalities provided by the Companion. It is important to recall that the *Companion* is integrated into the CoSKS interface.

Figure 4.25 shows the CoSKS user interface[20], which provides the CoSKS user with information about his/her Projects, Meetings, Tasks, Issues, Profile and a some CoSKS functionalities (*red box).*

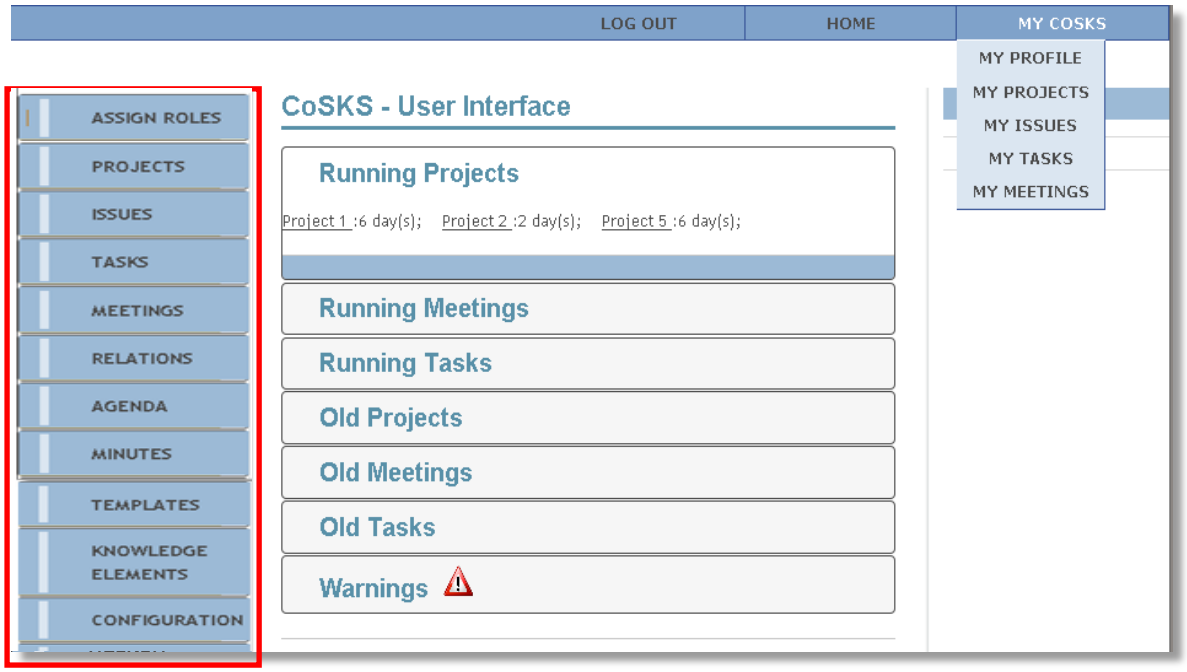


Figure 4.25 – CoSKS User Interface

For the sake of clarity, the legend bellow explains the two existing symbols in CoSKS User Interface.

- One day left to finish the Project, Task or Meeting.

- The Project, Task or Meeting finishes today.

The figures 4.26, 4.27 and 4.28 illustrate some important information and warnings given by *Companion* to the CoSKS user.

[20] The appendix C on page 95 shows an overview of the CoSKS user interface.

Figure 4.26 displays some information about projects (e.g. Project 1 will finish in nine days, Project 2 finishes today, and Project 5 will finish tomorrow), additionally Figure 4.27 shows the information about meetings, and finally Figure 4.28 illustrates the information about running tasks. It is important to refer that the information it is always related to the current user.

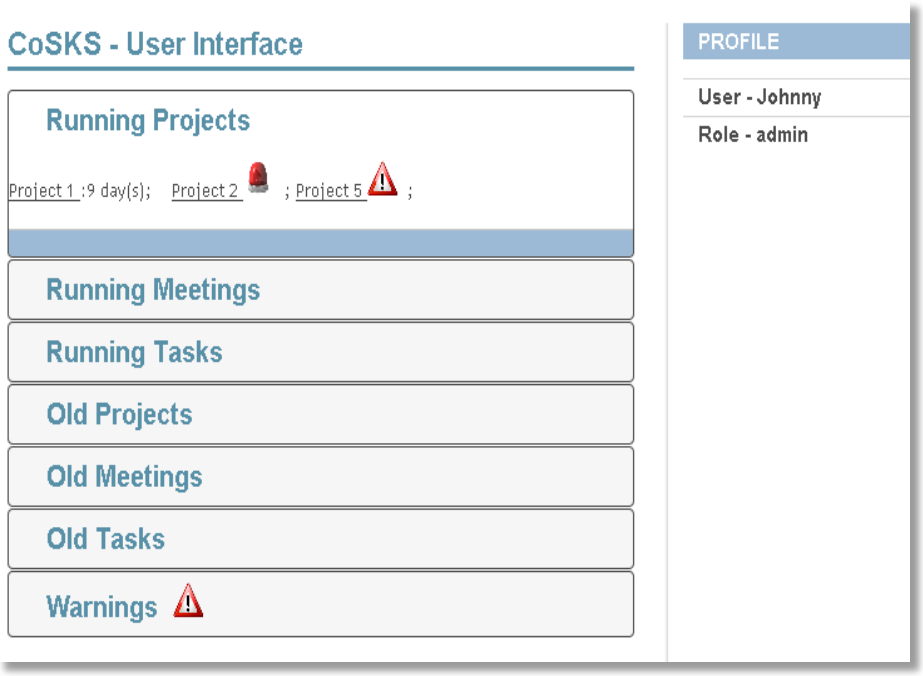


Figure 4.26 - User Interface (Project)

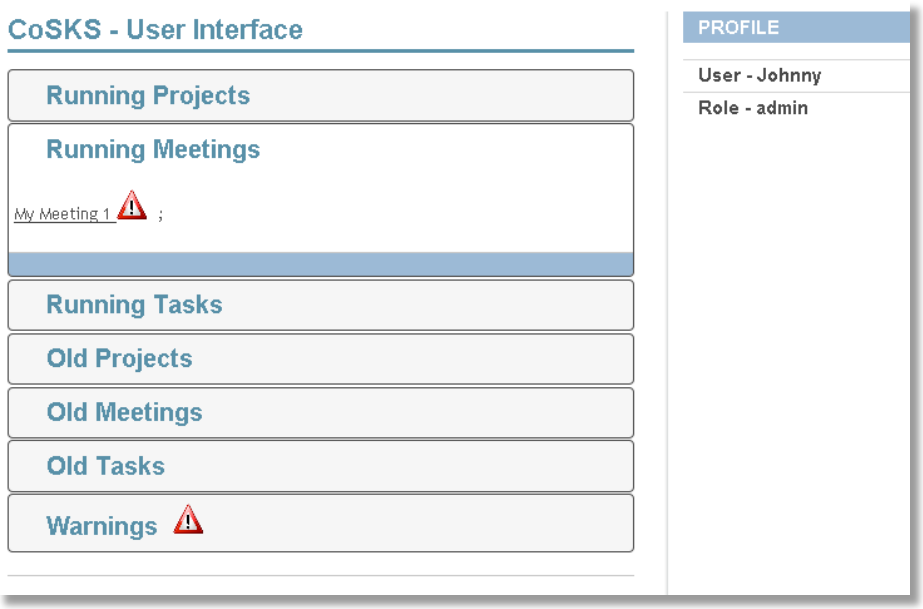


Figure 4.27 - User Interface (Meeting)

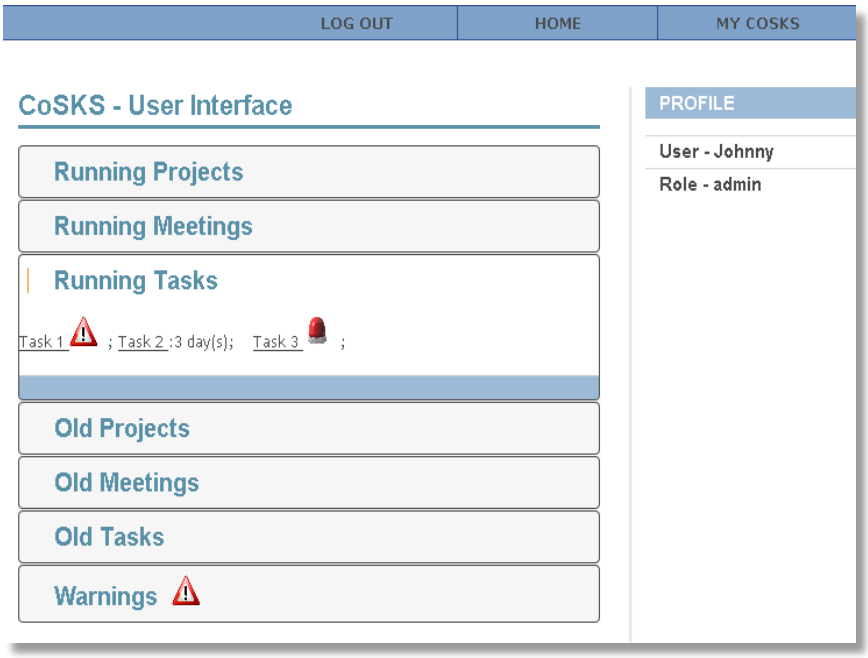


Figure 4.28 - User Interface (Tasks)

Figure 4.29 shows the *Warnings* section. This section displays all the projects, meetings, and tasks[21] that have reached their deadlines and have not been closed, yet. Even when K-Elems[22] reach their deadlines, they still have to be closed since *Companion* only extracts knowledge from closed projects, tasks, meetings, and issues.

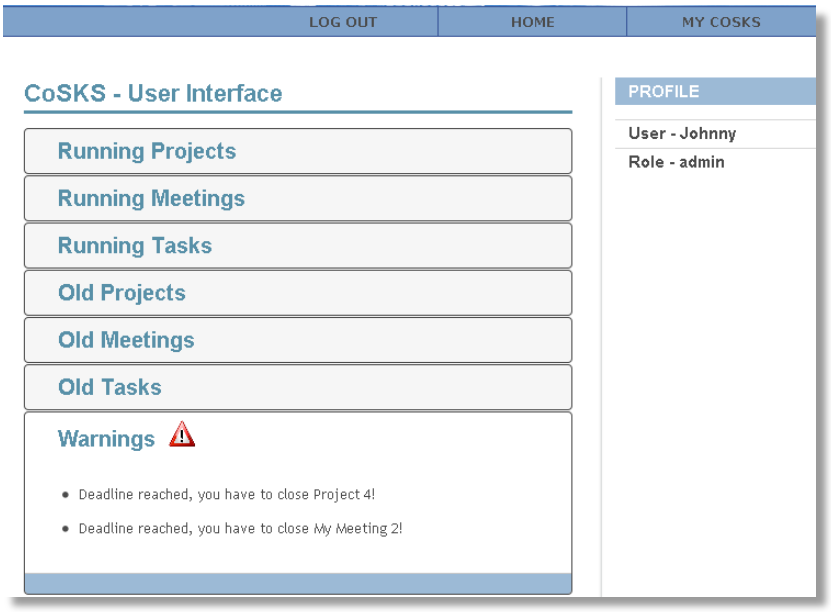


Figure 4.29 - User Interface (Warnings)

[21] Please see table 3.1on page 19 in order to remember the Knowledge Element (K-Elem) definition
[22] Projects, Meetings, tasks, and Issues.

By pressing the name of a K-Elem (in the user interface), *Companion* shows all the information about that K-Elem (description, possible problems, important documents to read and some forecasts).

Figure 4.30 and 4.31 are examples of how this information can be shown to the user. In the figure 4.30 it is possible to see the description of the selected K-Elem (in this example it is a project) and the figure 4.31 shows further information and help for that K-Elem.

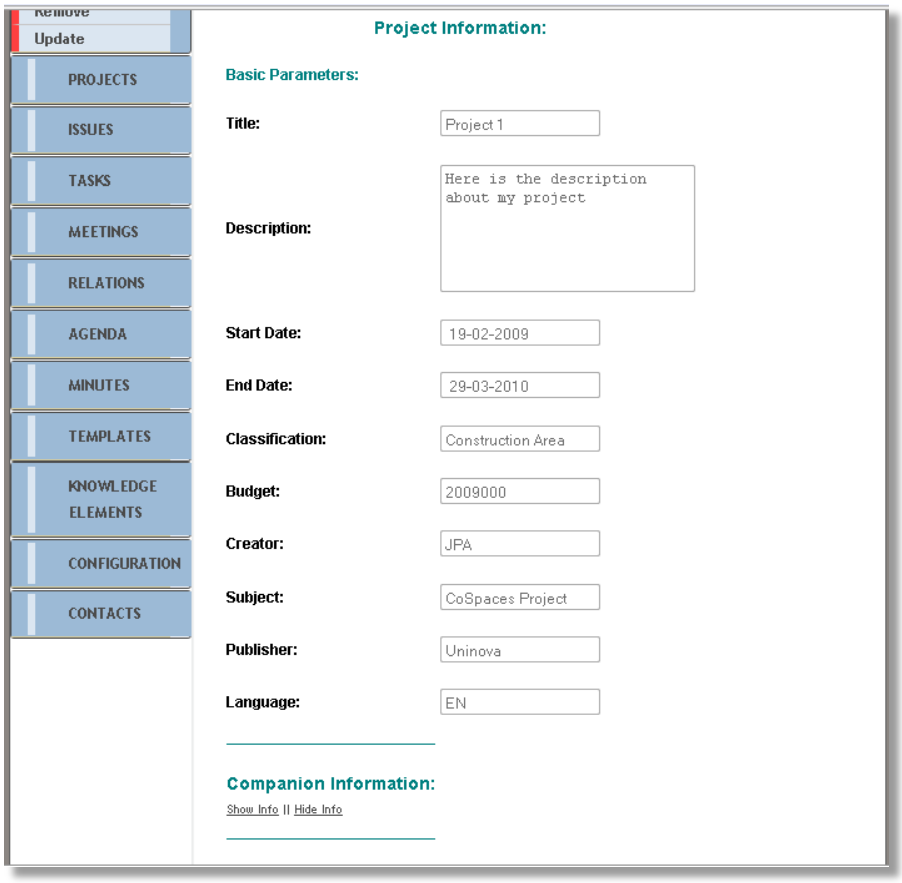


Figure 4.30 - Information About a K-Elem

Notice that *Companion* sorts and shows all the important documents and problems according to their relevance, for the selected K-Elem. (figure 4.31).

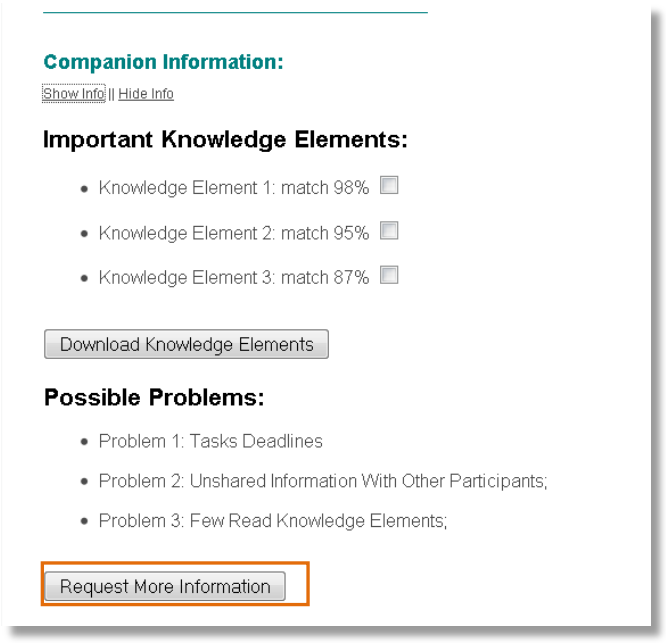


Figure 4.31 - Additional information about the selected K-Elem

As depicted on figure 4.31, there is a button named *Request More Information* (orange box). By pressing this button, the user can request reports[23] about similar and completed K-Elems.

When creating a project, task or meeting the *Companion* helps the user with relevant information. Figure 4.32 shows an example where three tasks were previously created and where the *Companion* provides the user with important information about them. The user can see that information by selecting one task and pressing the button *View Important Information For Task.*

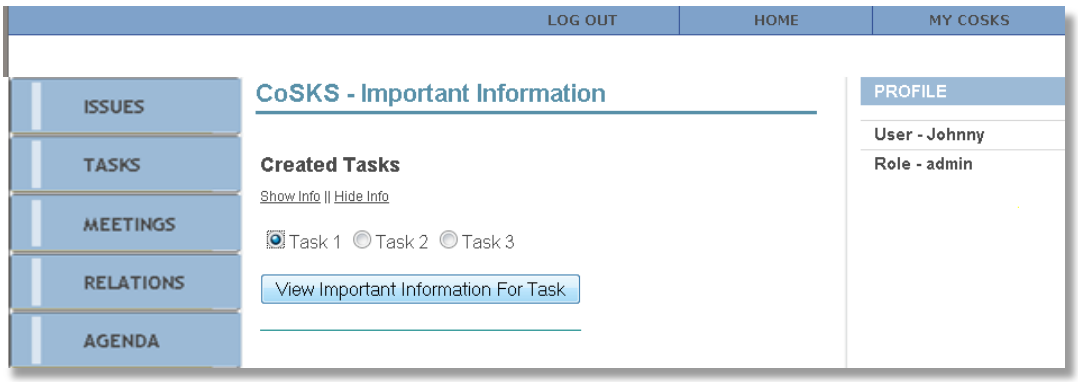


Figure 4.32 - Selection of a Task to See Important Information

[23] The Figure 4.37 on page 78 show an example of these reports.

After pressing the button *View Important Information For Task*, the user will see the page showed by figure 4.33. It has three different sections[24], namely: *Unsolved Issues*, *Actors Ranking*, and *Relevant Information About Similar and Previous Tasks.*

By pressing the label *Show Info* in each section, the user can have more information about the section.

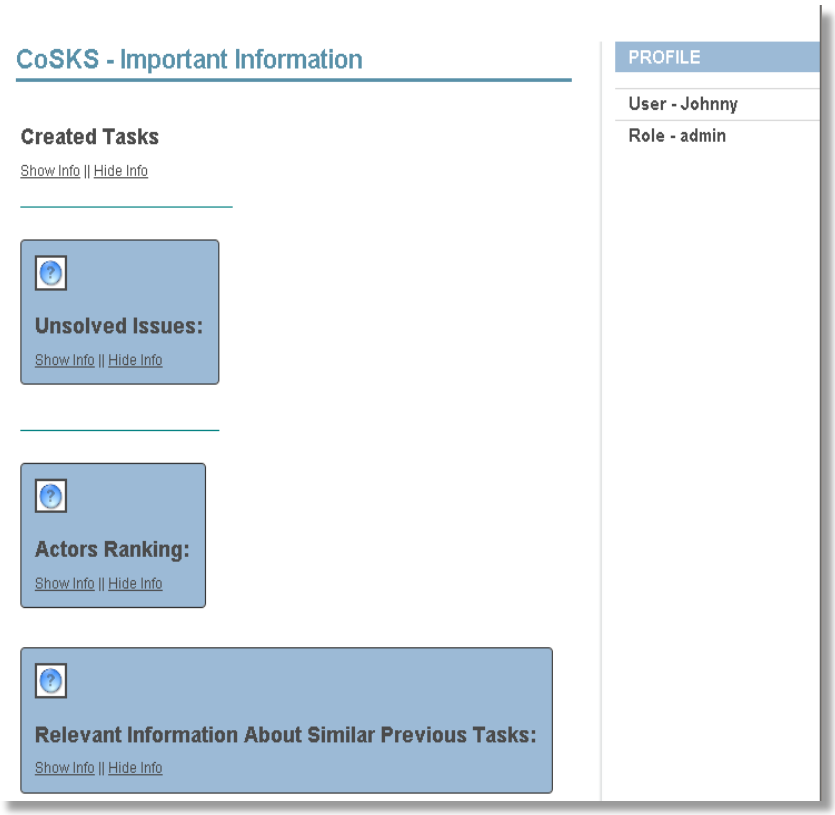


Figure 4.33 - Important Information for Chosen Task

Figure 4.34 and 4.35 show the possible information existing in both *Unsolved Issues* and *Actors Ranking* section.

[24] When this page appears the information of each sections is hidden.

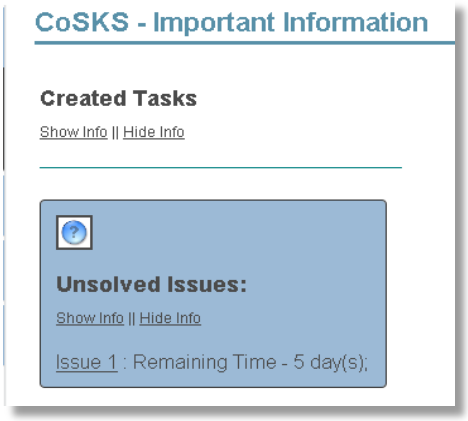


Figure 4.34 - Unsolved Issues Section

Figure 4.35- Actors Ranking Section

Notice that the actors are sorted according to the number of relevant documents that each actor has read or uploaded to the knowledge repository of the system (e.g. the actor Chris has 91 documents read or uploaded, more than any other actor, and for this reason is the first name of the list).

The figure 4.36 shows the information of the section *Relevant Information About Similar and Previous Tasks.* This section shows a list reports about similar and relevant finished tasks, i.e., each report corresponds to a finished task. These reports are sorted by their relevance to the selected task.

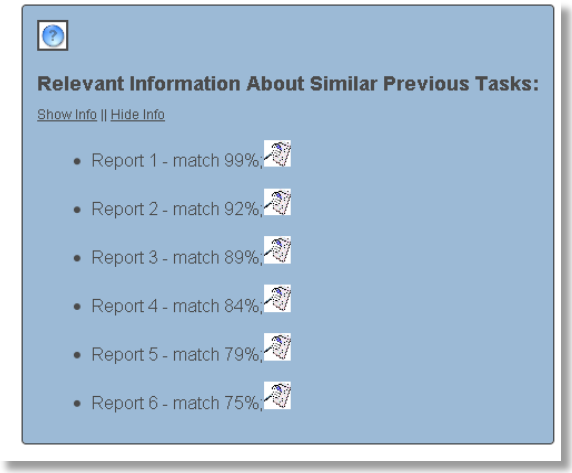


Figure 4.36 - Relevant Information About Similar and Previous Tasks section

The user can select one report, of one task, and see its information (Figure 4.37).

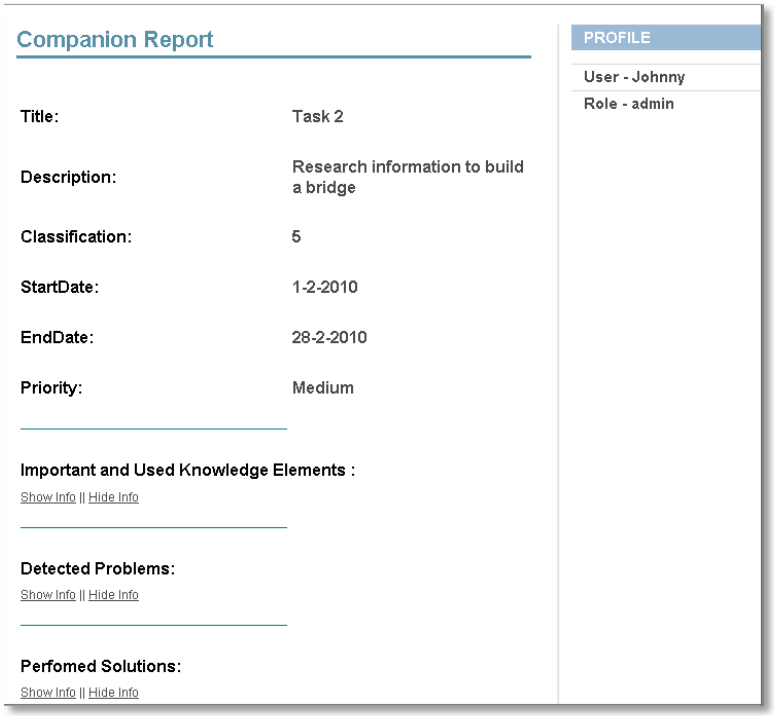


Figure 4.37 - Task Report

Figure 4.38 shows the information existing in the section, *Important and Used Knowledge Elements* (which shows the relevant documents read during the execution of the finished task).

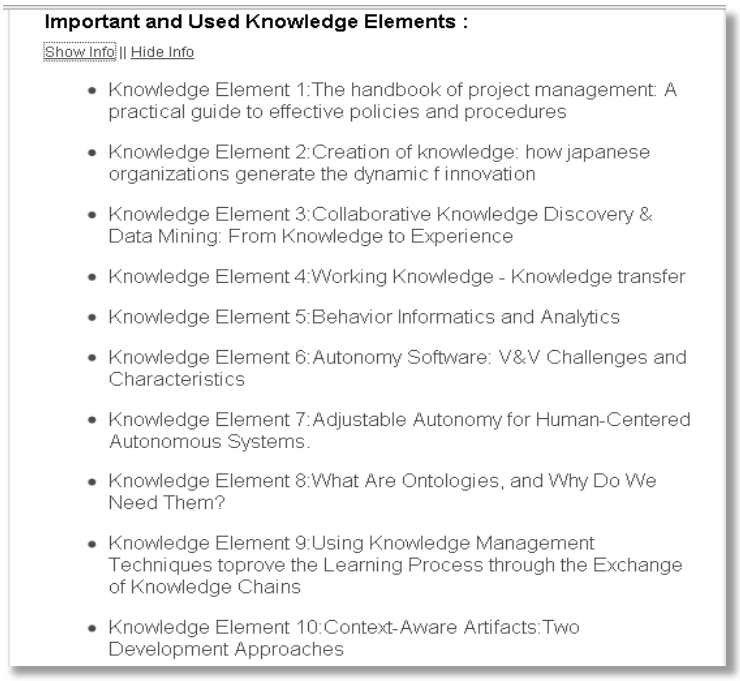


Figure 4.38- Relevant and Used Knowledge Elements Section

The figure 4.39 shows the *Detected Problems* section (that gets the problems found during the execution of the task) and the *Performed Solutions* section (which shows the solutions achieved to solve the problems of the performed task).

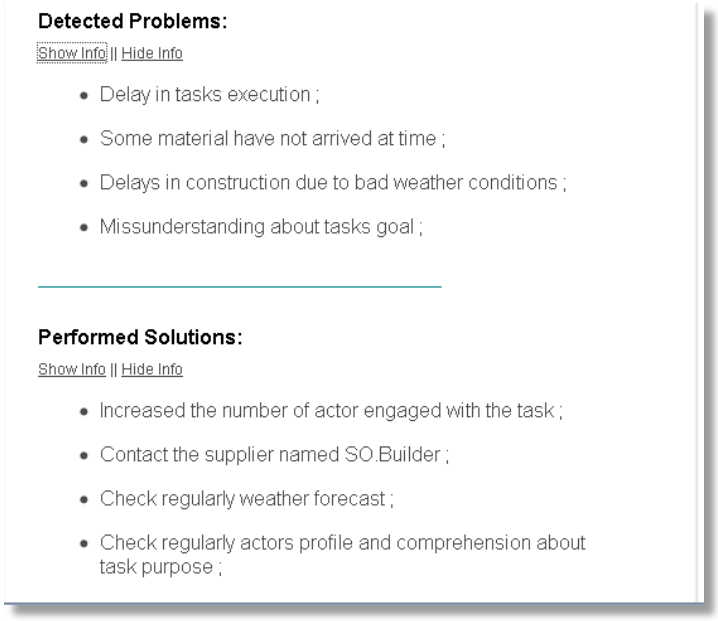


Figure 4.39 - Detected Problems and Performed Solutions Sections

When creating a meeting, the information and functionalities provided by the *Companion* are different than when the user is creating a task. Figure 4.40 shows an example where a

meeting was previously created and where the *Companion* provides the user with important information about it.

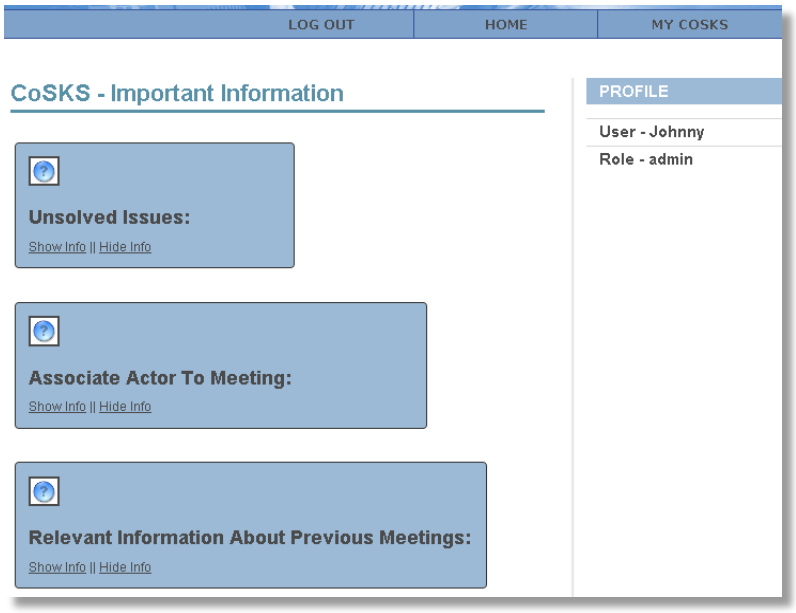


Figure 4.40- Important Information for Created Meeting

In order to see that information the actor has three different sections, namely:

- *Unsolved Issues* – This section shows all issues that are still unsolved. By selecting one or more issues and pressing the button *Create Agenda for Issues*, it is possible to create automatically an agenda for each selected issue (figure 4.41)

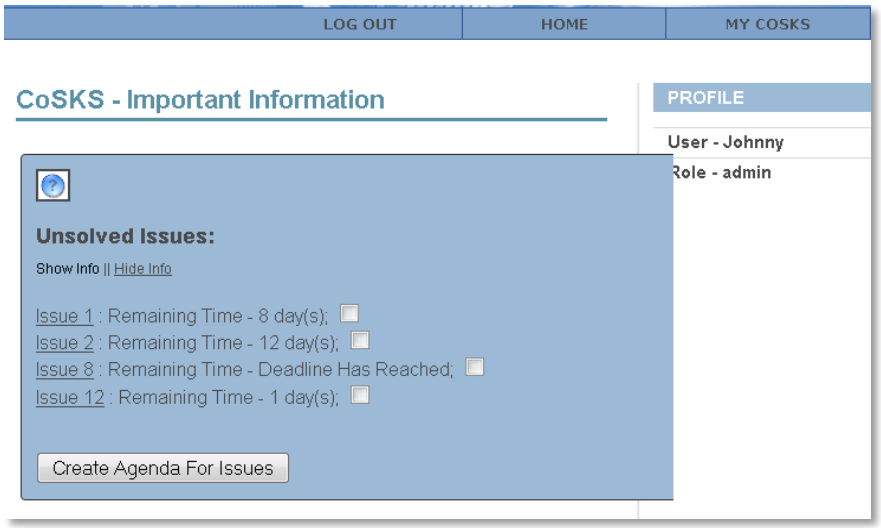


Figure 4.41 - Unsolved Issues Section

- *Associate Actors to Meeting* – This section shows all actors that work on similar meetings. The user can click on the name of each actor and automatically associate him/her with the meeting (figure 4.42)

Figure 4.42 - Associate Actors to Meeting Section

- *Relevant Information About Similar and Previous Meetings* – The information in this section is similar to Figure 4.36 on page 74.

Figure 4.43 shows an example where an issue was previously created and where the *Companion* provides the user with two different options, namely:

- *Add Tasks* – Where user can create some tasks to solve the issue.
- *See Important Information* – Where *Companion* shows some sequence of tasks that have solved similar and finished issues.

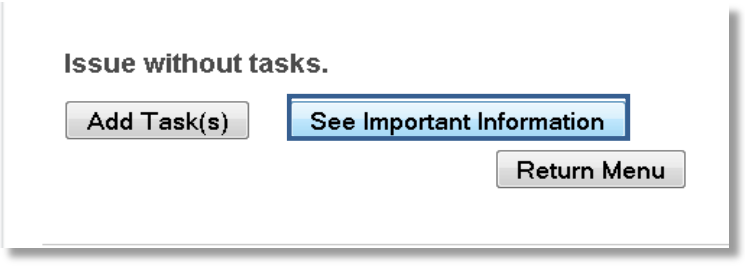


Figure 4.43 - Two Possible Options after Creating an Issue

By pressing the button *See Important Information* (figure 4.43) the user will see the page showed by figure 4.44. This page displays the information about the created issue and offers the option to see some possible sequences of tasks to solve the issue (*Possible sequence of tasks to solve this issue*).

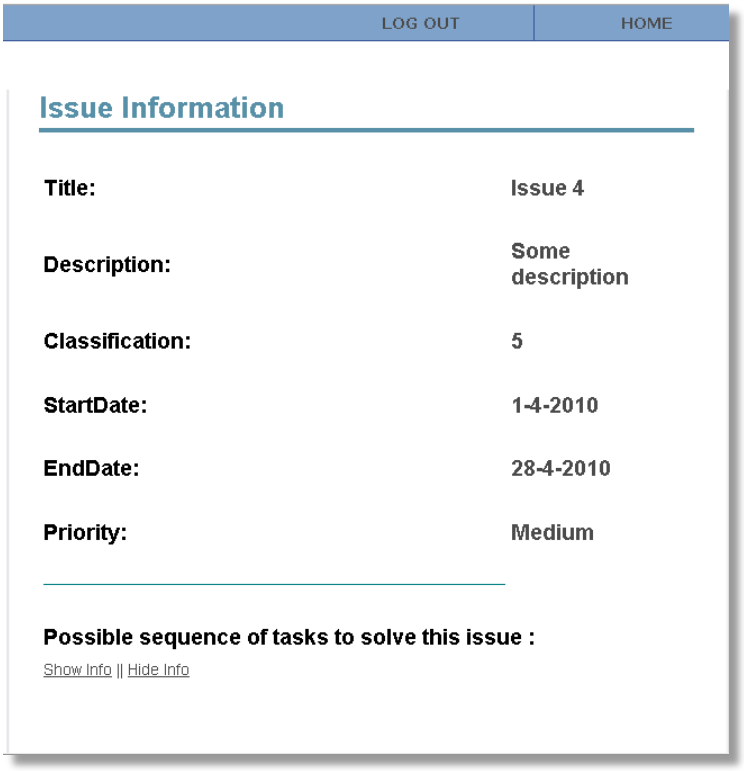


Figure 4.44 - Issue Information and Sequence of Tasks to Solve it

Figure 4.45 shows the potential information existing in the section *Possible sequence of tasks to solve this issue.* This section offers several reports containing some sequences of tasks that can solve the created issue.

Those sequences of tasks are sorted by their relevance to the created issue and by the likelihood of solving it properly.

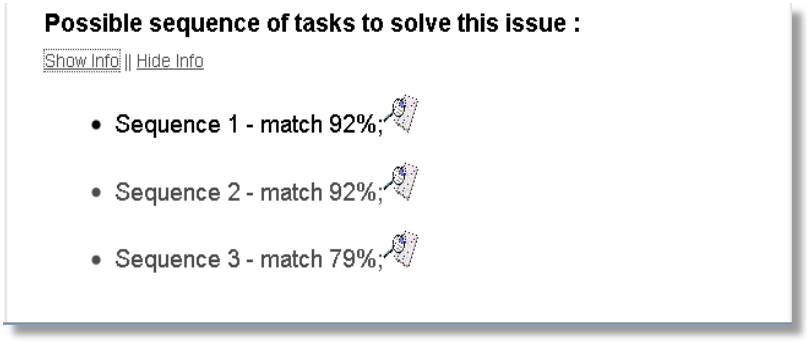


Figure 4.45 – Possible sequence of tasks to solve this issue Section

By selecting one sequence of tasks, the user will see a page similar to figure 4.46. This page shows some possible sequence of tasks to solve the created issue. Notice that it is possible to select the issue name or a tasks name, in order to see its information.

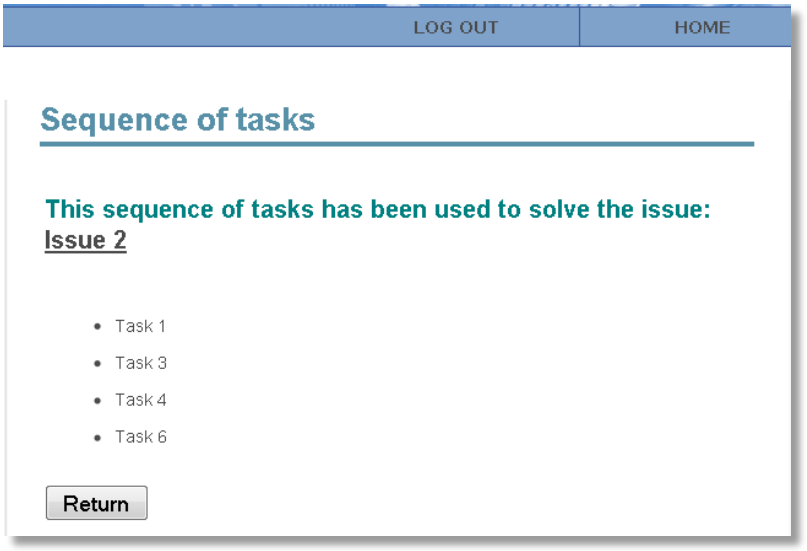


Figure 4.46 - Possible Sequence of Tasks to Solve the Issue

Chapter 5 – Conclusions

This chapter presents the final conclusion of this work. It starts with a short summary about the developed work and an explanation of the achieved goals. Additionally it presents the contribution of this research and the open issues and the future work to be developed.

5.1 Synthesis of the Work

This work designed and implemented a component called *Companion* aiming to help and improve the management level of projects, within a Collaborative Workspace.

Companion is materialised through the CoSKS component, which was developed to handle the knowledge dimension of the CoSpaces project. From a conceptual perspective, CoSKS considers that engineering projects go through a series of Decisional Gates, where issues are raised and/or decisions (targeting the appropriate solutions for those issues) are made. Between two decisional gates, CoSKS adopts a proactive behaviour that plays a fundamental role when addressing the collaboration aspect within a project oriented scenario.

Users need to create, disseminate and capitalize knowledge, and they must be awarded with tools that can effectively support them in their daily tasks. Such tools must establish, in a seamlessly way, the right interfaces between user and business requirements. This work foresees that self-awareness systems, implemented by proactive and reactive services are capable of reducing such barriers towards an effective collaboration environment.

This work has accomplished the following goals:

- ***Observe all processes involved in a project (e.g. tasks, issues, meetings, actors selection, milestones, and deadlines) and detect possible deviations or problems that could jeopardize the project's success***. *Companion* is able to detect problems or possible deviations with its proactive and autonomous behavior.

- ***Provide a constant surveillance and monitoring of the work that is being carried out.*** With the context-awareness capability, *Companion* can detect all the environmental changes and monitoring all the existing work.

- ***Offer a set of solutions to solve a particular problem or a pertinent and appropriate document to help the user, always regarding the performance and efficiency enhancement.*** Supplied by Data-Mining and Semantic Services, when *Companion* detects a problem (using its proactive behavior and context-awareness capabilities) it requests to the Data-Mining services some possible solutions or important documents to help the user with his work. These Data-Mining services sort the solutions and the documents, according to their importance and efficiency.

- ***Offer personalized information to each user, according to their features.*** By knowing which user is logged and by analyzing his/her profile and access rights, *Companion* offers individual and personalized information to each user (using Data-Mining and Semantic services as well).

- ***Create an adaptable, open, flexible and scalable component, in order to make it an easy pluggable component.*** With two possible configurations, *Full Services Configuration* and *Integrated Miner Services Configuration*, *Companion* can be connected to any existing system (with or without Data-Mining and Semantic services).

5.2 Contribution of the Research

This work offered a theoretical and practical understanding of the complexities existing in the implementation of a software infrastructure that provides the Collaborative Workpaces with special behavioral features, supported by KM and reasoning concepts.

Within the context of the CoSpaces Integrated Project, this thesis adopted, proposed, and used several important concepts, namely: *Decisional Gates, Context-Awareness, Semantic Vectors, Proactivity, Reactivity, Autonomy*, *Data Mining and Ontologies*.

One of the major concerns in this work was to develop an easily pluggable component, in order to be connected with different systems. With has an adaptable, open, flexible and scalable architecture this component can provide any Collaborative Workspaces with particular behavioral features, even in legacy systems.

During the development of this dissertation the following papers were written:

1. Costa, R., Lima, C., Antunes, J., Figueiras, P., & Parada, V. (2010). Knowledge Management Capabilities Supporting Collaborative Working Environments in a Project Oriented Context. *2nd European Conference on Intellectual Capital*, (pp. 1-9). ISCTE Lisbon University Institute, Lisbon, Portugal and Polytechnic Institute of Leiria, Portugal.

2. Lima, C., Costa, R., Maló, P., & Antunes, J. (2010). A Knowledge-based approach to support decision making process in project-oriented collaboration. To appear in *11th European Conference on Knowledge Management.* Vila Nova de Famalicão, Famalicão, Portugal.

5.3 Open Issues & Future Work

Due to financial and time constrains, it was not possible to test the *Companion* with, cameras, microphones, sensors, etc., in order to assess the full range of context-awareness capabilities during a meeting.

As future work, there are still some important points to be improved, such as:

- Adoption of model generated workplaces approaches coupled with self-aware systems.

- Improvement in order to get better and more accurate context-awareness skills.

- The adaptability of such a component should be tuned to the user's profile in terms of his/her behavior patterns and his/her ways of handling interruptions.

- Improve the architecture, in order to be more efficient and reliable.

References

[1] Rezende, J., & Souza, J. (2007). Using Knowledge Management Techniques toprove the Learning Process through the Exchange of Knowledge Chains. *IEEE* , pp. 681-686.

[2] Rubio, J. M., & Crawford, B. (2008). An approach towards the integration of adaptive business intelligence and constraint programming. *IEEE* , pp. 364-369.

[3] Zhang, N., & Lu, W. F. (2007). A Framework for Managing Enterprise Knowledge for Collaborative Decision Support. *IEEE* , pp. 517-519.

[4] Dalkir, K. (2005). *Knowledge Management in Theory and Practice.* McGill University: Elsevier.

[5] Choi, J. H., Kang, D. H., Jang, H., & Eom, Y. I. (2008). Adaptive Access Control Scheme Utilizing Context Awareness in Pervasive Computing Environments. *IEEE* , pp. 491-498.

[6] Oyama, K., Jaygarl, H., Xia, J., Chang, C. K., Takeuchi, A., & Fujimoto, H. (2008). Requirements Analysis Using Feedback from Context Awareness Systems . *COMPSAC* , pp. 625-630.

[7] Loke, S. W. (2006). Context-Aware Artifacts:Two Development Approaches. *PERVASIVE computing* , pp. 48-53.

[8] Bao, Y., & Xianyi, Q. (2009). Research of Business intelligence which based upon web". *IEEE* , pp. 1-8.

[9] Stijn, V. (2008, December). Linking Business Intelligence into your business. *IT Pro* , pp. 28-34.

[10] Zhao, L., & Huang, X. (2009). Research on the application of business intelligence in logistics management. *IEEE* , pp. 1-8.

[11] Zhang, L., & Tu, X. (2009). A feasible Enterprise Business Intelligence Design Model. *IEEE* , pp. 182-187.

[12] Fong, S., & Chan, S. (2002). Mining Online users' Access records for Web Business . *IEEE* , pp. 579-762.

[13] Seufert, A., & Schiefer, J. (2005). Enhanced Business Intelligence-Supporting Business Processes with Real-Time Business Analytics. *16th International Workshop and Expert Systems Applications* , pp. 1-8.

[14] Lima, C., Costa, R., Maló, P., & Antunes, J. (2010). A Knowledge-based approach to support decision making process in project-oriented collaboration. To appear in *11th European Conference on Knowledge Management.* Vila Nova de Famalicão, Famalicão, Portugal.

[15] Wehmeier, S. (Ed.). (2002). Oxford Advanced Learner's Dictionary (p.714, 6th ed). New York: Oxford University Press.

[16] Cao, L. (2008). Behavior Informatics and Analytics: Let Behavior Talk. *IEEE International Conference on Data Mining Workshops*, (pp. 87-96). Sydney, Australia.

[17] Dorais, G. A., Bonasso, R. P., Kortenkamp, D., Pell, B., & Schreckenghost, D. (n.d.). Adjustable Autonomy for Human-Centered Autonomous Systems. *IEEE* , pp. 1-20.

[18] Schumann, J., & Visser, W. (2006). Autonomy Software: V&V Challenges and Characteristics. *RIACS / NASA Ames* , (pp. 1-6). Moffett Field, CA,.

[19] CHEVERST, K., BYUN, H. E., FITTON, D., SAS, C., KRAY, C., & VILLAR, N. (2005, June 5). Exploring Issues of User Model Transparency and Proactive Behaviour in an Office Environment Control System. *User Modeling and User-Adapted Interaction* , pp. 1-39.

[20] Lima, C., Malo, P., & Costa, R. (2009). Knowledge Support For Collaborative Workspaces: The CoSpaces Approach. *CIB-W102*, (pp. 1-10). Rio Janeiro.

[21] Chandrasekaran, B. and Jorn R. Josephson, V. Richard Benjamins. 1999. What Are Ontologies, and Why Do We Need Them? IEEE Intelligent Systems. 14 (1): pp. 20 - 26.

[22] NetBeans IDE 6.8 Release Information, Accessed 2010-05-01 at http://netbeans.org/community/releases/68/

[23] Relational Persistence for Java and .NET, Accessed 2010-05-01 at http://www.hibernate.org/

[24] About Apache Struts 2, Accessed 2010-05-01 at http://struts.apache.org/2.x/index.html

[25] About MySQL, Accessed 2010-05-01 at http://www.mysql.com/?bydis_dis_index=1

[26] JavaServer Pages Technology, Accessed 2010-05-01 at http://java.sun.com/products/jsp/

[27] Apache Tomcat, Accessed 2010-05-01 at http://tomcat.apache.org/

[28] Visual Paradigm, Accessed 2010-05-01 at http://www.visual-paradigm.com/

[29] Davenport, T., & Prusak, L. (2000). Working Knowledge - Knowledge transfer. *Harvard Bussiness Review* , pp. 88-106.

[30] Horeis, T., & Sick, B. (2007). Collaborative Knowledge Discovery & Data Mining: From Knowledge to Experience. *IEEE* , pp. 421-423.

[31] Ko, K.-E., & Sim, K.-B. (2008). Development of Context Aware System based on Bayesian Network driven Context Reasoning Method and Ontology Context Modeling. *International Conference on Control, Automation and Systems* , (pp. 2204-2231). Korea .

[32] Nonaka, I. (1994). The Dynamic Theory of Organization. *Journal of Organization* , 14-37.

[33] Nonaka, I., & Takeushi, H. (1997). *Creation of knowledge: how japanese organizations generate the dynamic f innovation.* Rio Janeiro: Campus Editor.

[34] Salomie, I., Anghel, I., Cioara, T., & Dinsoreanu, M. (2008). A Context Awareness Model Enhanced with Autonomic Features . *IEEE* , pp. 239-246.

[35] So, R., & Sonenberg, L. (2004). Situation Awareness in Intelligent Agents: Foundations for a Theory of Proactive Agent Behavior. *IEEE/WIC/ACM International Conference on Intelligent Agent Technology* , (pp. 1-7). Australia.

[36] Trad, A., Kalpic, D., & Fertalj, K. (2002). Proactive Monitoring of the Information System Risk and Quality . *24th Int. Conf. Information Technology Interfaces IT1*, (pp. 279-284). Cavtat, Croatia.

[37] Young, T. L. (2006). The handbook of project management: A practical guide to effective policies and procedures. London: Kogan Page.

Appendix A- UML Class Diagram

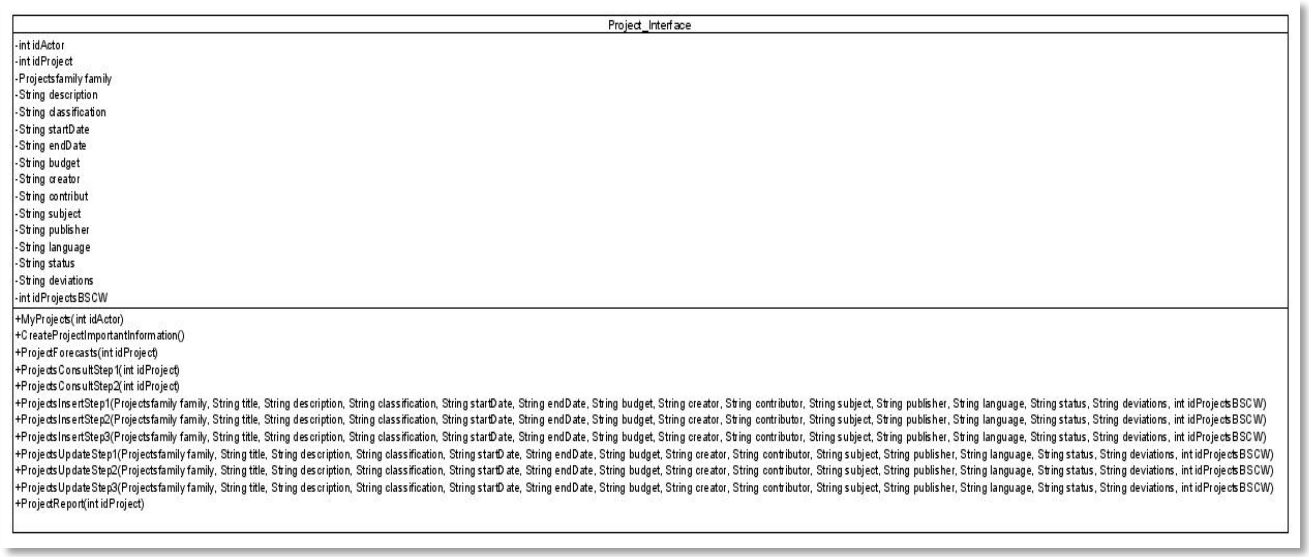


Figure 1 – Project Interface

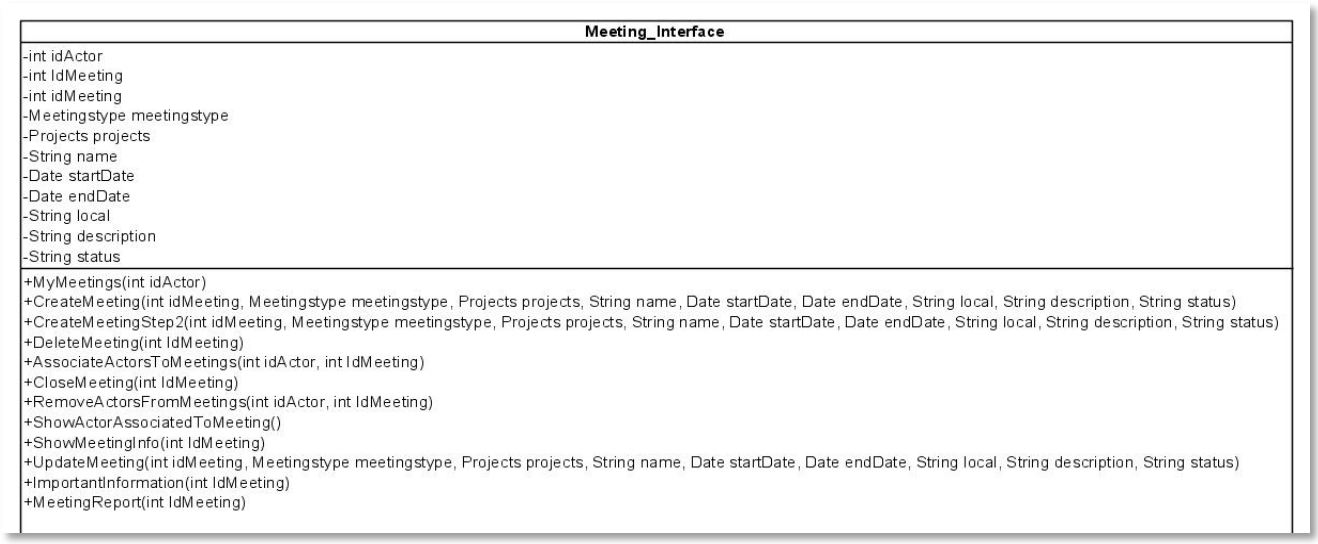


Figure 2 – Meeting Interface

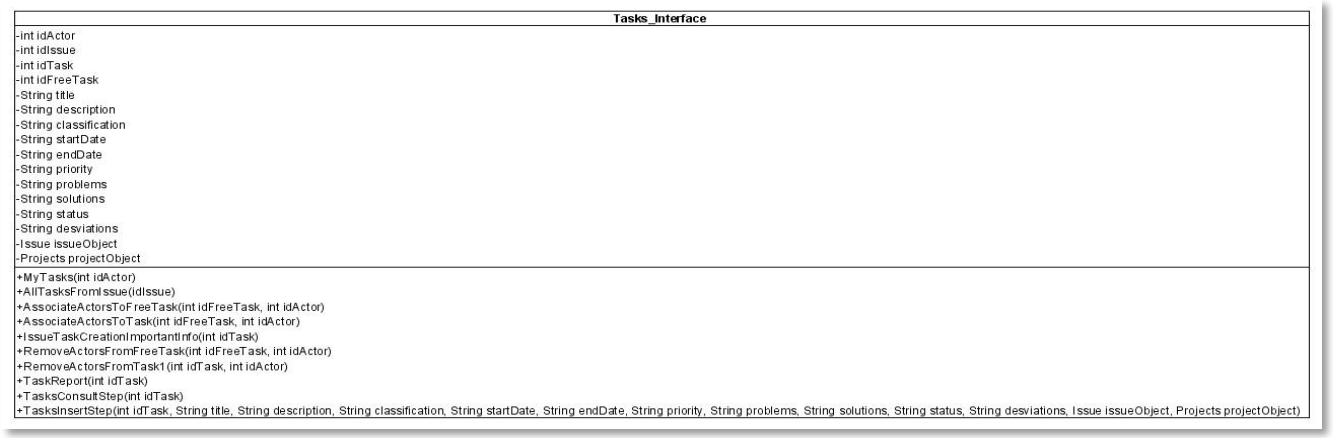


Figure 3 – Task Interface

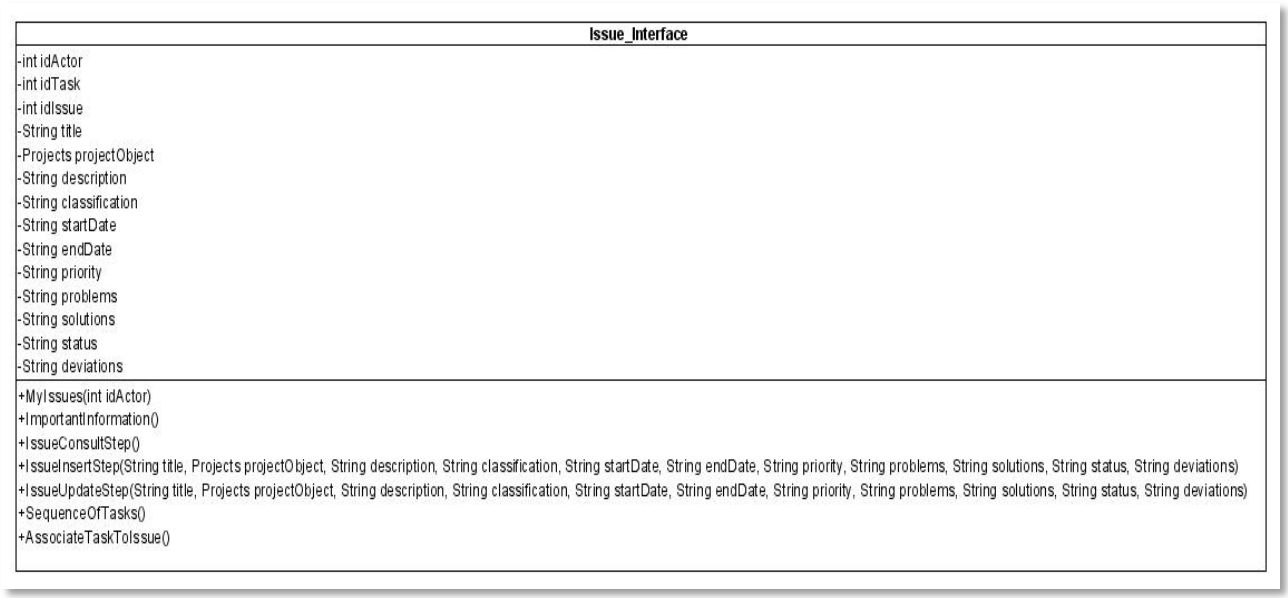


Figure 4 – Issue Interface

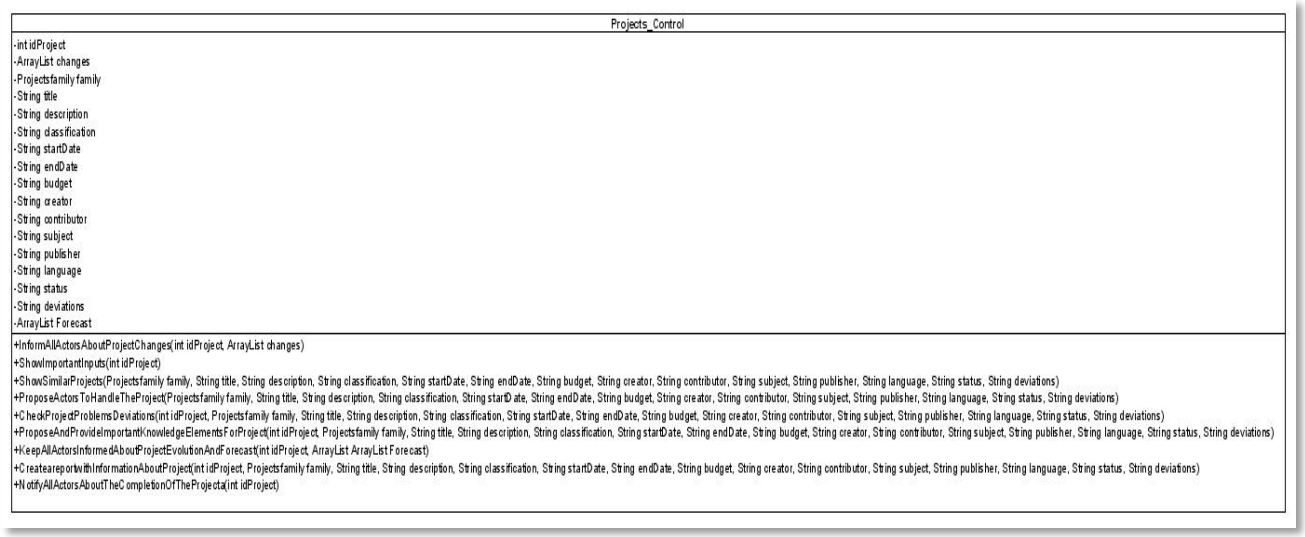


Figure 5 – Project Control

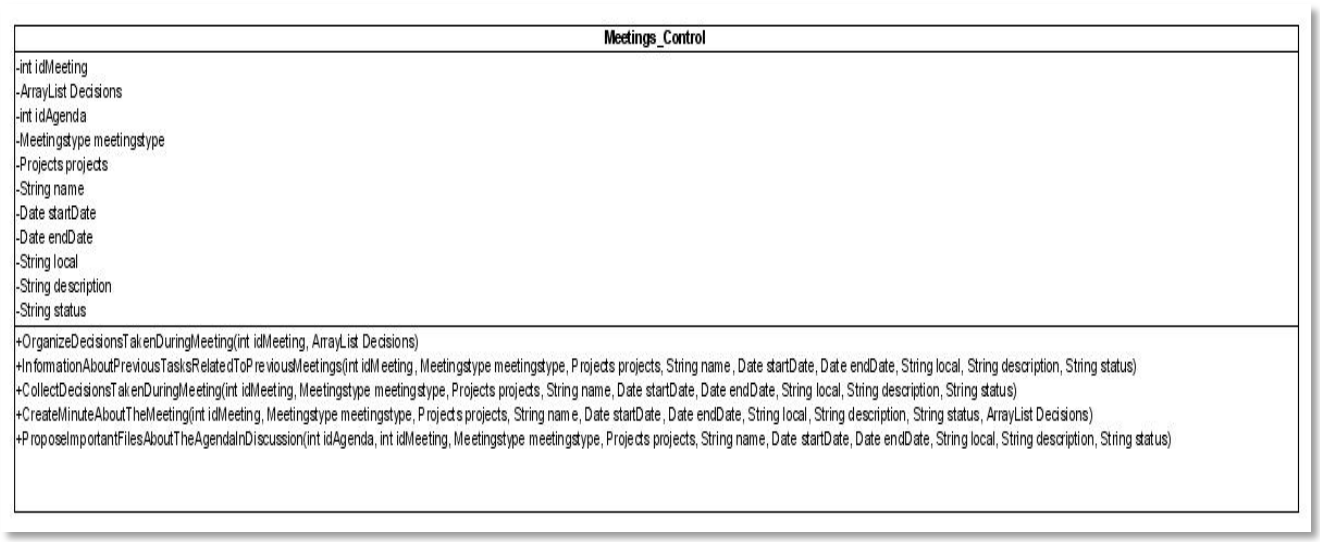


Figure 6 – Meeting Control

Figure7 – Task Control

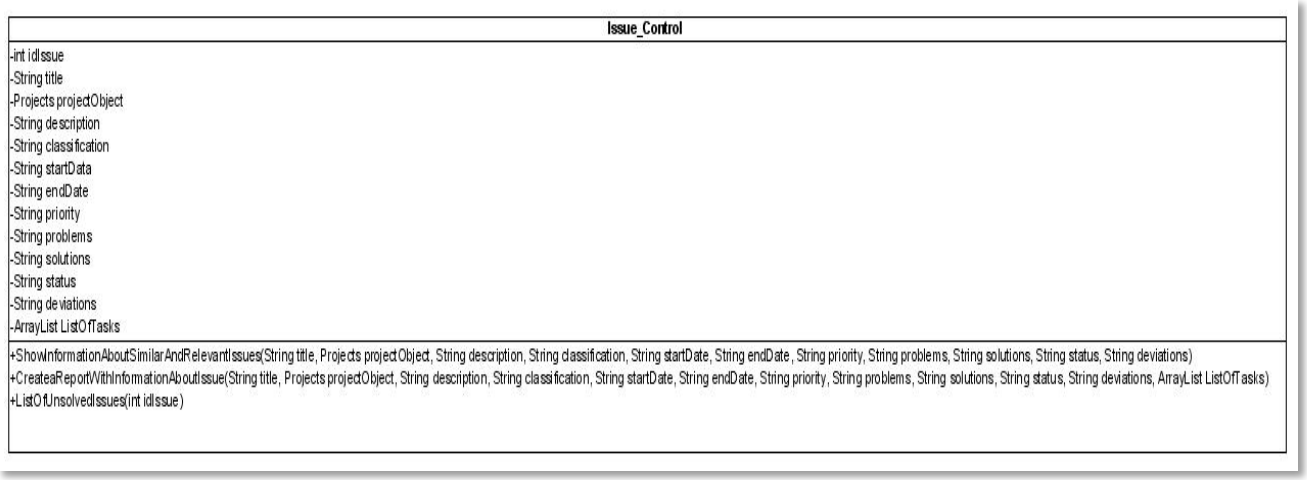


Figure 8 – Issue Control

Figure 9– Project Entity

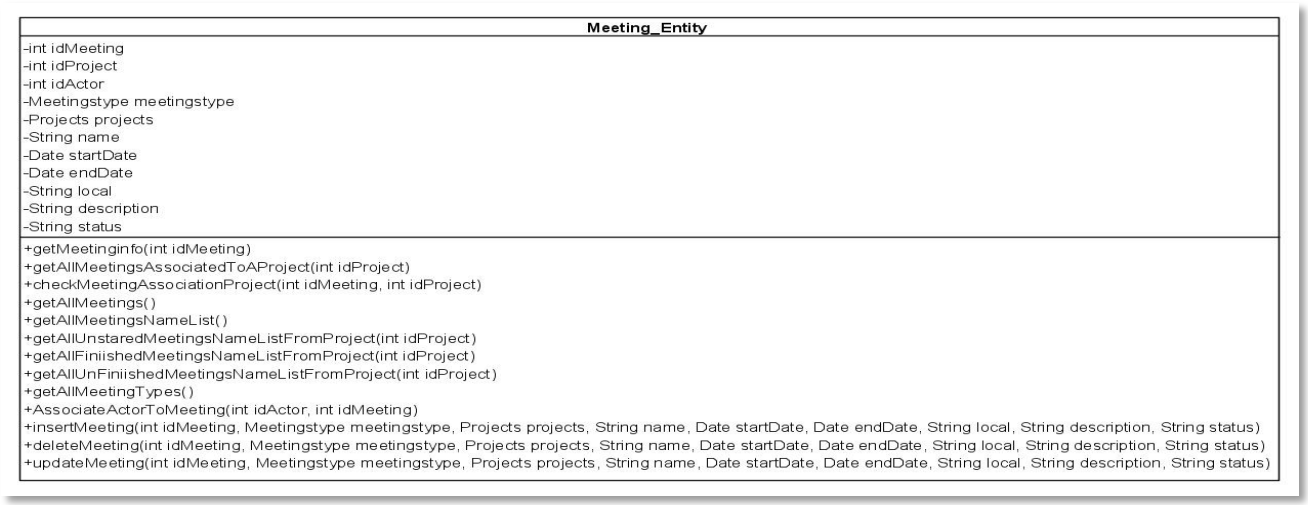


Figure10– Meeting Entity

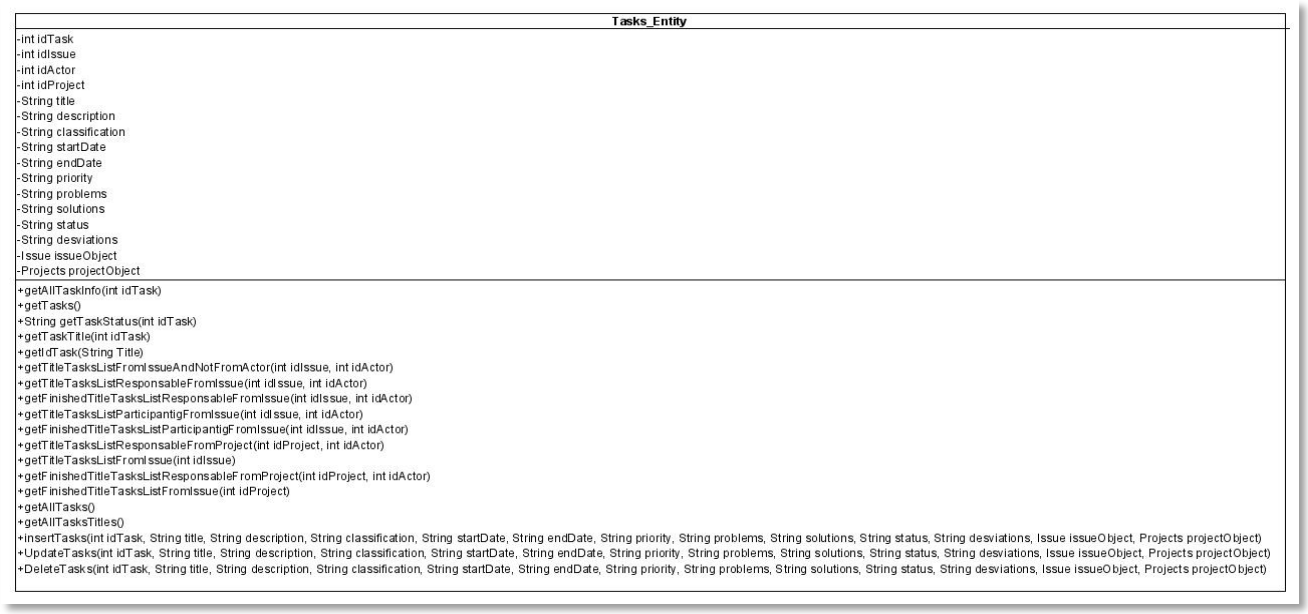


Figure 11– Task Entity

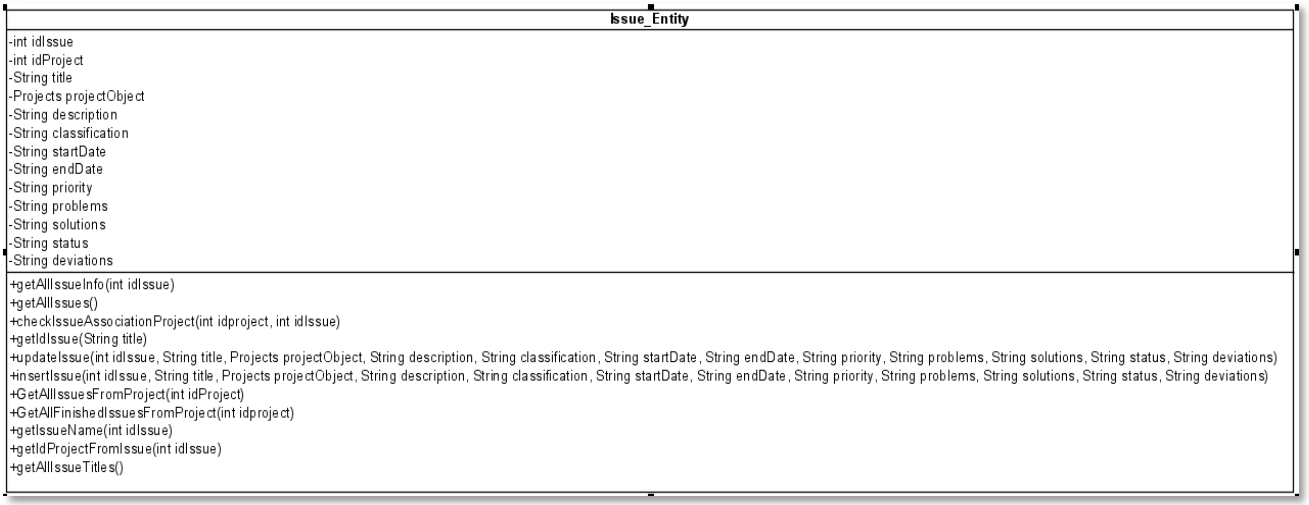
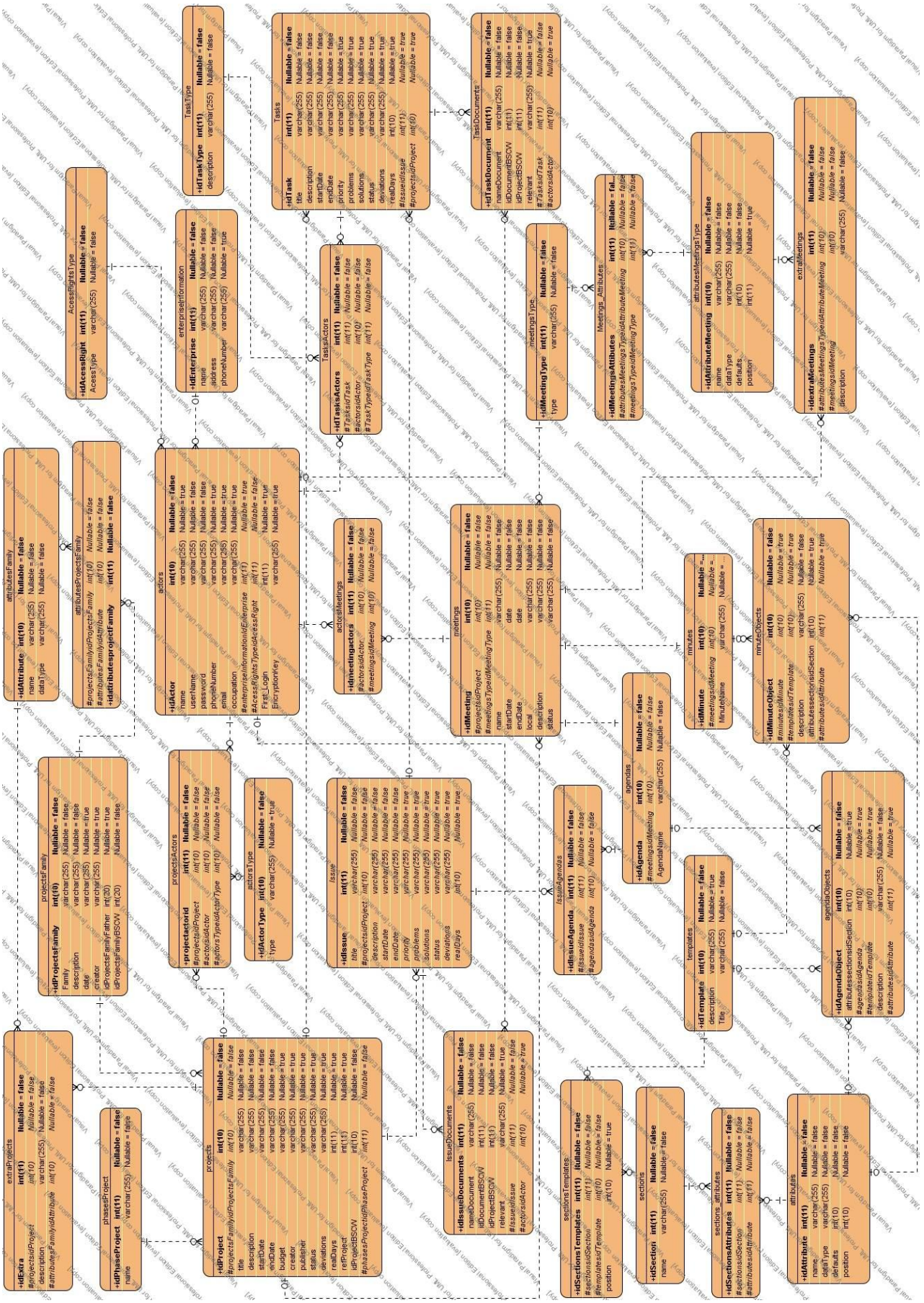


Figure 12– Issue Entity

Appendix B-Entity-Relationship Diagram



Appendix C- CoSKS User Interface