Collaboration intelligente et transparente. Apports de l'informatique ubiquitaire au travail collaboratif assisté par ordinateur.

Kahina Hamadache

UNIVERSITE LILLE 1

ECOLE DOCTORALE SCIENCES POUR L'INGENIEUR

Doctorate of Philosophy
in
Computer Science

Kahina HAMADACHE

SMART & SEAMLESS COLLABORATION

Bringing Pervasive Computing
to the Computer Supported Collaborative Work

Thesis supervised by Luigi LANCIERI

Defended on November 30th, 2011

Committee:
Chairman: Lionel SEINTURIER
Reviewer : Bertrand DAVID
Reviewer : Serge GARLATTI
Examiner : Nadine LUCAS
Guest : Stephan GOUALIER

Acknowledgement

First of all, I would like to express my deepest gratitude to my supervisor, Pr. Luigi Lancieri. I owe him a great debt for his supervising and also for giving me the freedom to pursue my own curiosity.

I appreciated very much the way I have been welcomed by Stephan Goualier at France Telecom and for all the support I have received. I like to thank Patrice Manson who I have shared office with. I also like to extend my thanks to my colleagues Samuel Legouix, Alain Ottavi, Olivier Gruson, and Mathieu Lecouvey.

Thanks to Bertrand David and Serge Garlatti for their time and their constructive comments. Thanks to Lionel Seinturier for accepting the chair of the jury. Thanks to Nadine Lucas for her support and to represent the University of Caen.

Thanks to Mr Marc Gazalet, head of the SPI Doctoral School, for having accepted me for my last year.

Great thanks to Yvan for his support and for his way to permanently make me give the best of myself to always go further and beyond.

I would like also to thank my best friend Kelly for helping me relax, go out, have fun, dance and enlighten my mood.

No words could express how much I am indebted to my family. Jeddi, Vava, Yemma, Lydia, Sonia, Massi and Amara thank you for all your support and love during these years.

Abstract

This thesis work contributes both in the pervasive computing and computer supported collaborative work domains. We explore these domains by an extended presentation of related works concerning models, designs and evaluation methods. Our main contribution for these areas is the PCSCW model, which proposes an original approach to the integration of the pervasive aspect inside the collaboration. By relying on an ontological model representing users' context and a set of devices collaboration rules, our work enables smart devices to analyse their context and find the best way to behave and collaborate with other devices of the environment in order to seamlessly and efficiently channel and facilitate the collaboration of humans. We also propose a methodology allowing collaborative systems' developers to build their own evaluation strategies.

Keyword: Collaborative Work, Computer Supported Collaborative Work, Context Awareness, Context Representation, CSCW, CSCW Evaluation, Evaluation Method Taxonomy, Evaluation Strategy, Ontological Model, OWL, PCSCW, Pervasive Computing, Pervasive Computing Simulator, Pervasive Computing Supported Collaborative Work, Reasoning Rules, SWRL.

Résumé

Ces travaux de thèse apportent une contribution aux domaines de l'informatique pervasive et du travail collaboratif assisté par ordinateur. Nous explorons ces domaines par une présentation étendue de différents travaux se rapportant aux modèles de collaboration, aux différentes conceptions ainsi qu'aux méthodes d'évaluation. Notre principale contribution pour ces domaines est le modèle PCSCW, qui propose une approche original pour l'intégration de l'aspect pervasif au sein de la collaboration. En se basant sur un modèle ontologique représentant le contexte des utilisateurs ainsi que sur un ensemble de règles de collaborations entres machines, notre travail permet aux dispositifs intelligents d'analyser et de trouver la meilleure façon de se comporter et de collaborer avec les autres machines de l'environnement afin de canaliser et de faciliter de manière transparente et efficace la collaboration entres les humains. Nous proposons également une méthodologie permettant aux développeurs de systèmes collaboratifs pervasifs de construire leurs propres stratégies d'évaluations.

Mots clés: Conscience du contexte, Evaluation des TCAO, Informatique pervasive, Modèle ontologique, OWL, Règles de raisonnement, Représentation du contexte, Simulateur d'informatique pervasive, Stratégie d'Evaluation, SWRL, Taxonomie de méthodes d'évaluation, TCAIP, TCAO, Travail collaboratif, Travail collaboratif assisté par l'informatique pervasive, Travail collaboratif assisté par ordinateur.

Content

List of figures

Introduction

In our society no domain has known such a rapid growth as Information Technology. For the past sixty years Computer Sciences has evolved from hangar-sized computers to perform simplistic calculations to small devices taking an invading part in everybody's life allowing them to listen their favourite music online, chat with people on the other side of the planet in real time, make movies, and so on. In fact, this domain has taken such a place in our world that it's almost impossible to think that we could one day step backward and simply renounce to use it.

Firstly intended to be a tool for research and military purposes, the real life of Computer Supported Collaborative Work Systems has started with the rapid adoption of computers in companies and the development of business oriented applications. One of the main reasons of this growth is the evolution of users. Indeed, if the use of computers was once reserved to highly qualified scientists or Information Technology experts, it has become more and more accessible and almost everyone can now access and use a computer. This accessibility has spawned new generations accustomed to use computers and smart devices. This habit has made the adoption of new technologies and computing patterns far faster and more efficient.

Another critical point in the development of computer supported collaborative work is the revolution of Internet. The Web has weaved links between peoples in a way that had never been anticipated. This has been done not only by supporting and facilitating classical communications means such as mail and voice but also by allowing new ones like instant messaging, video conferencing, document sharing, online playing, etc. Thus, the possibility for humans to share artefacts whether they be texts documents, pictures, videos, webpages, games situations or whatever you may want to share with others has dramatically improved interactions and communications.

Obviously, computer supported collaborative work has not been avoided by this evolution and has rapidly taken advantages from it. The first natural benefit of Internet for collaborative work is that you can contact anyone on the planet as soon as he has access to an active connection. Moreover, with the development of web services and more advanced online applications it is now possible to create, manage, share and publish documents with nothing more than a web browser. Despite these incredible benefits, Internet is not spared by common network issues, a major one is the recurring problem of security, security of data transmitted in the case of professional needs, privacy of persons when personal data can be intercepted by hackers, security of interactions, etc. Even if this problem can't be perfectly solved, all networking problem aren't unbearable, for instance the hard problem of concurrent editing can be solved by different manner such as document locking or versions comparison. Globally, benefits of Internet for collaborative work greatly overcome the implied difficulties. Computer supported collaborative work can sound like a simple extension of single user computing for people who don't actively participate to the development of such systems. Still once you've stepped in this domain, your first impressions rapidly fade and you realize that multiplying users implies not only one, but several additional dimensions to the classical theory. Thus, CSCW introduce new difficulties of assorted kinds: design and development difficulties, ethnographic difficulties (users' usage of electronic devices and software can rarely be predicted and the addition of computers between two, or more, humans can lead to unexpected collaboration patterns). Besides, evolutions of information technologies bring even more complexity in this field. For instance, while some years ago the preferred pattern to design collaborative application was the classical client-server method where there were only few servers and a constellation of clients that had to connect to servers, current researches tends to promote the peer-to-peer model where each user in the network is client and server. This relatively new pattern is powerful but poses some problems when it comes to clearly organize the collaboration.

Scope of thesis

This thesis is situated at the intersection of several domains. The first one is Computer Supported Collaborative Work (CSCW). As we have already mentioned this domain has known important mutations in the last decade as it has been made available for mass and thus has dramatically extended its field of experimentation and its feedbacks, allowing it to evolve even faster. Besides, there are other factors of evolution in this domain: growth of computer performances, large increases of network bandwidth allowing people to have Internet connection comparable to what people had as local network connection some years ago, development of wireless connection is also a major point in this evolution, allowing people to be connected in their preferred places (hotel, school, office, train station, airport, library, etc.).

The second domain being part of our interest field is mobility. This domain covers in fact a wide area of perspectives as it can be mobility of persons, mobility of equipment, mobility of networks, social mobility, etc. Indeed, a person can stood physically still while moving from an Ethernet network to a wireless one. On the other way, you can be moving between two campus of your university and still being connected to the same network (even if it's not the same access point). Another mobility we don't always see as it, is the one taking place when a user switches from its professional computer to his personal one, in that case the user hasn't changed, his physical location has not changed but the computer he's using is probably not the same, and he may not have access to the same networks.

The third domain involved in our research is context management, it covers several aspects: context awareness involving context modelling, context information collecting and interpretation, context monitoring and context adaptation. Those aspects imply different challenges, in particular in term of architecture as context management can be hard and costly to deploy. Another particularly complex task in the context management process is the interpretation of data to restore the meaning of actions at a computational level, this interpretation is required to be able to efficiently reason over the context and then adapt the behaviour of applications.

The fourth and last domain we were interested in is Pervasive Computing. This field is relatively new; it is based on the intelligent communication of devices of the environment to provide information and services adapted to user's context. As context information types are various, the adaptation can be done according to one or several context aspects. As an example of pervasive services we can site location based services which adapt the behaviour of applications according to the current location of the user.

Situated at the intersection of collaborative work, mobility, context management and pervasive computing our work has focused on the exploration of the interactions between these fields and the improvement of these interactions. A particularly critical point in our research has been the intersection of collaborative work and pervasive computing and how it was possible to combine them. As we'll see in the chapter dedicated to this part, there is very few work that have been done on this particular aspect and we think our work could be the beginning of further researches.

Before considering the problem itself, it is important to point out the fact that our work doesn't focus on some specific pervasive applications; instead we adopt the point of view assuming that the pervasive computing paradigm has to be integrated into lower layers and designed to be generic. Naturally, it is not the only way to deal with the consideration of Pervasive Computing coupled with collaboration work, we may have focused on the development of rich services, providing users with detailed information relative to the collaboration. However our deepest thoughts are that by starting from a lower level and by providing some native abilities, we may create new basis to build on, allowing unexpected patterns and opportunities to appear.

Problem statement

Collaborative work in its absolute consideration has predated humans for millions of years if we consider animals such as social insects. Considering collaboration between humans, it has to be as old as our kind. Computer supported collaborative work is born with "modern" computing some decades ago and has closely followed its evolution. For the past few years information technologies are evolving toward the multiplication of smart electronic devices such as smartphones, laptops, TVs, GPS and so on. Despite or maybe because of this proliferation the digital environment is a non-continuous space where miscellaneous devices can communicate, or not, with others. Thus, in order to make this space "continuous" the Pervasive Computing is based on the communication between devices to smartly adapt their behaviour to the current context of users and offer them a seamless interaction with the digital world. The continuity problem is a particular point that attracted our attention. Indeed, no matter what kind of service or resource a device can offer, if a discontinuity exists in its way to accomplish its task, it may not be able to do it. Such discontinuities can be of various types: a classical "gap" in a network if a gateway machine goes offline, a lack of availability (for instance if a server is overloaded of requests), a discontinuity of information update leading to outdated and potentially inconsistent data.

If we refer to **[Yenumula Venkataramana Reddy, 2006],** Pervasive Computing is set to progress in four directions: Computing, Communication, Cognition and Collaboration, the 4Cs of Pervasive Computing. They point out that for a truly pervasive system the 4Cs are unavoidable. For now, 2 of the 4Cs are in advance: Computing and Communications. The Cognition depicts the need for a real "context awareness" of systems, allowing them to "understand" a situation and adapt their behaviour. Toward this goal, Artificial Intelligence can provide most of the trails to develop such system (like expert systems, case based systems and else). Last but not least the Computer Supported Collaborative Work area is forming the last 'C'. As we already mentioned, CSCW tries to facilitate the cooperation of humans in their everyday work by providing them tools, structures and services. Despite the importance of collaboration for Pervasive Computing, it is a particularly poorly considered aspect in the pervasive computing literature.

We have quickly reached the same conclusions as **[Yenumula Venkataramana Reddy, 2006]** and then decided to focus our work on the four aspects we already evoked: collaborative work, mobility, context management and pervasive computing.

The main underlying question of our work is: How can we bring pervasive computing to computer supported collaborative work? How can we facilitate the work of humans through pervasive computing? From the beginning of our work, one thing was obvious: this research field is too large to be treated in only three years; hence we chose to focus on some specific points that could give us a complete view on this domain. Our starting point was: how can we efficiently model the collaboration in pervasive computing systems and how can it improve their behaviour. More precisely, how can we ensure, or at least improve the continuity of the digital space through the collaboration of smart devices and finally facilitate the collaboration of users by optimizing the usage of this space. Our exploration of the literature on this subject showed that it has been loosely considered and that only few works really developed such model. As a natural continuation we were interested in the evaluation of computer supported collaborative work systems in pervasive environment. However as the evaluation of CSCW offered no real consensus we restrained our investigation to "traditional" CSCW systems with the objective to find a method allowing the evaluation of simple systems as well as complex pervasive systems.

Summary of contributions

Several contributions can be noticed in this dissertation, among which we can underline the following major ones:

- **Pervasive Computing Supported Collaborative Work (PCSCW)**:

- ✓ ***PCSCW Model***: in order to improve the integration of collaborative work aspects with pervasive computing and make them benefit from each other we propose a model that allows describing users with their roles, tasks, actions and resources required to perform them. Then by comparing required resources to available ones we can trigger device cooperation routines to facilitate the collaboration of users through the continuity of service and automation of simple actions.

- ✓ ***Device Collaboration rules***: design, development and use of devices collaboration rules for the PCSCW model. These rules relies on the precise description of roles, tasks, actions, resources required by these actions and constraints associated to these resources to select the proper way to make devices cooperate with the final objective to facilitate the collaboration of humans. We suggest that by defining constraints on resources as triplets composed of a parameter, a value and an associated criticality it allows us to quantify, estimate, compare and then choose between several candidate rules. The finality given by these rules is a simple but efficient way to make devices choose automatically the most appropriate way to cooperate.

- ✓ ***PCSCW Simulator*:** designing a model is important, but if you have no way to evaluate and validate it, it's almost a loss of time. In that perspective we developed a simulator for the PCSCW model based on the JADE framework (Java Agent DEvelopment) that helped us to validate the model by testing different device cooperation routines. In addition to the validation of the model itself, this simulator offers the opportunity to develop and evaluate cooperation rules and then find new "pervasive behaviours".

- **Pervasive CSCW Evaluation**:
  - ✓ ***Taxonomy and strategy of evaluation***: our research in the CSCW evaluation field took us to the reading of numerous articles which gave us the opportunity to notice that the number of different methods to evaluate collaborative work systems is outrageous. As we wanted to take advantages of this profusion, we propose to organize evaluation of CSCW systems in Evaluation Strategies that provide the evaluation process for a given system by defining what kind of methods to use at the different steps of this process. These strategies are based on a taxonomy of evaluation methods and the description of evaluation context. With this solution we intend to be able to plan almost any evaluation of CSCW system.

  - ✓ ***Evaluation Indicators*:** we address the evaluation of computer supported collaborative work taking place in a pervasive environment. More precisely we focus on the search and definition of indicators allowing the evaluation of such systems. These indicators allow evaluators to quantify different aspects of a collaborative system which are: simplicity, effectiveness, reliability, security and quality. They also constitute a base supporting the comparison between two different systems and different settings of the same system.

- **Pervasive surveying service**: to immerse ourselves in the pervasive computing domain we started our research by the development of a pervasive service intended to enhance the collaboration between scattered users by providing them an efficient platform to summarize their opinion through the use of Semi Open Multiple Choices Questionnaires. This service also relies on the interaction between mails and forums. This allows users using non ergonomic devices to efficiently collaborate.

Dissertation overview

This dissertation is divided in three parts. In the first part, we present a state of the art concerning the main notions we adopted for our work in order to give us the relevant and necessary background. In chapter one, we cover a large sample of the past and recent literature on computer supported collaborative work. For the second chapter we present different approaches helping model the CSCW. Third chapter describes researches in the evaluation of CSCW systems. Fourth and last chapter of this part we propose a focus on the foreseeable evolutions of the CSCW.

In the second part of this dissertation we develop our contributions in the computer supported collaborative work area. Fifth chapter and first of this part, we expose how we developed a new collaborative service combined with a pervasive facet, the pervasive surveying in situation of mobility. Sixth chapter, we introduce our Pervasive Computing Supported Collaborative Work Model that provide a simple way to model the collaboration between users and the resources required to perform collaborative tasks. Third chapter of this part, we formalize our work about CSCW with a taxonomy of evaluation methods and the introduction of evaluation strategies. After that we detail the design of the simulator we developed for our PCSCW model. Final chapter of this part, we present the results of the evaluation campaign we led on our work.

Discussion and perspectives on our work constitutes the third part of this dissertation and allows us to put back our contribution in perspective with our hopes and the needs we had considered.

Part 1: Research Background

Chapter 1 Computer Supported Collaborative Work overview

Chapter 2 Modelling Computer Supported Collaborative Work Systems

Chapter 3 Evaluating Computer Supported Collaborative Work Systems

Chapter 4 Evolution of CSCW

Chapter 1 CSCW overview

In this chapter we review and provide the context for the research reported in this dissertation. Our review includes analysis and discussion over the main aspects of CSCW systems. We present the already existing collaboration models for CSCW based on tasks, roles and awareness. Then we review the evaluation of such systems, in particular we point out the fact that evaluation is a critical need but has been left untreated for too long. We also analyse the evolution of CSCW systems and focus on some interesting part: the context awareness which is still a recent aspect of this domain, mobile CSCW and Pervasive Computing. Finally we give some trails on the next possible evolutions of the CSCW systems. This overview will serve as background and further reference for the rest of chapters.

1.1 Origins

Living in a social environment, people's everyday activities, including work, study, and play inevitably involve collaboration with others. The idea of supporting collaboration with computer can be traced back to the 1960, when Douglas Engelbart **[Engelbart and English, 1968]** realized the first collaborative system: oNLineSystem. Although it was the first collaborative system, NLS was an incredibly advanced system, supporting many of the basic functions of modern groupware such as e-mail, shared annotations, shared screens, shared pointers, and audio/video conferencing. This work was intended to allow Engelbart's team spread throughout the United States to write a paper on several hands.

In 1984, Iren Greif and Paul Cashman coined the term "Computer-supported Collaborative Work" at a workshop attended by individuals interested in using technology to support people in their work **[Grudin, 1994]**. Since then, CSCW has been interpreted and understood in a number of different ways. Some researchers use the term to express the idea of collaboration among a group of people using computers **[Howard, 1988]; [Kling, 1991]**. To others, CSCW represents "the design of computer based technologies with explicit concern for the socially organized practices of their intended users" **[Sushman, 1989]**, **[Hughes et al, 1991]**, the focus here is more loosely on the sociality of work, rather than on specific characteristics of collaborative work as a distinct category of work. Still, others, especially Scandinavian systems developers, emphasize participatory design **[Clement and Van Besselaar, 1993]; [Greenbaum and Kyng, 1991]**. CSCW is also referred to as "software for groups of people" or "groupware". This term is often used by people who focus on the design of software that support group work.

Bannon and Schmidt **[Bannon and Schmidt, 1989]** define CSCW as: "CSCW should be conceived as an endeavour to understand the nature and characteristics of cooperative work with the objective of designing adequate Computer-based technologies". This definition of CSCW combines an understanding of how people work in groups and how computer networking technologies can be designed to support activities. CSCW systems are collaborative environments that support dispersed working groups so as to improve quality and productivity. In 1992, Schmidt and Bannon restate their position, and identify several important questions, listed below, which they believe CSCW researchers must answer.

1. What characteristics distinguish cooperative work from individual work, and what support requirements derive from those characteristics?
2. Why do people work together, and how can computers be applied to address the requirements arising from the specific reasons?
3. How can coordination requirements arising during cooperative work be accomplished more easily using computer technology?

4. What do the identified requirements imply for the development of system architectures and services?

It is recognized that CSCW is inherently a multi-disciplines each contributing a different set of skills.

- Social Sciences (Sociology, theory of organizations) taking into account the organization of the persons, their contributions, the efficiency of a group;
- Distributed artificial intelligence, cognitive sciences for the interpretation of semantics of the information, the planning, the help to the realization of tasks in common;
- Human computer interaction, for the conception of interfaces multi-user, and the use made by users;
- Distributed computing, distributed systems and networks for the storage, the transfers and the exchange of information.

On the one hand, many authors consider that CSCW and groupware are synonyms. Ellis, in **[Ellis, 1993]**, defines groupware as "computer based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment". On the other hand, different authors claim that while groupware refers to real computer-based systems, CSCW focuses on the study of tools and techniques of groupware as well as their psychological, social, and organizational effects. Wilson's **[Wilson, 1991]** expresses the difference between these two concepts: CSCW is a generic term, which combines the understanding of the way people work in groups with the enabling technologies of computer networking, and associated hardware, software, services and techniques.

1.2 Services and applications

The first IT tools helping individuals to collaborate at work have been computers, office applications and shared storage media, followed afterwards by local office networks enabling mainly files transmission and shared repositories. In recent decades, the fast development of computer hardware and software technologies, especially the expansion of internet, has boosted research on CSCW. There are numerous kinds of systems that have been developed to support different kinds of collaborative interactions. In this section we outline some of the main types of system that figure in the literature.

1.2.1 Computer supported collaborative editing

Computer Supported Collaborative Editing is an important branch of CSCW, and it is one of the most active research areas. Collaborative editing systems allow users to view and edit the shared document at the same time and at different time from geographically dispersed sites connected by communication networks.

A number of collaborative editing systems have been developed. The aim of the different systems varies, some are specifically supporting collaborative authoring, some are general purposeful text editor, some support collaborative sketching or drawing, and some provide a framework for integrating existing editors into a collaborative environment.

Google Docs: Google Docs[1] is a free web-based shared workspace (editor) for group editing using bulleted lists, sorting by columns, adding tables, images, comments, formulas, changing fonts, using charts, and so on. Documents constructed in Google Docs are owned by the author, who can invite others to collaborate on authoring, reviewing or editing. Read-only access can also be granted. The editing environment is familiar; it looks like Word processing

[1] Google Docs: http://docs.google.com

programs that people have used before (like Microsoft Word or Open Office). Google Docs is a sufficient office suite for anyone wanting to quickly make some documents, for example if you want to make a spreadsheet of your monthly budget and you want to share it with your husband. It offers a really simple interface in the same style as other Google pages.

Another editor ***SubEthaEdit***: SubEthaEdit[2] (first known as Hydra) is a powerful collaborative text editor designed for Mac OS X that uses features from Bonjour (formerly called Rendezvous), Apple's discovery service. Unlike versioning systems such as Subversion or CVS, which keep a consistent copy of a document, SubEthaEdit provides collaboration with the ability to edit the same document in real-time, together with everyone in a group. SubEthaEdit is particularly suited for Extreme Programming and collaborative note-taking in conferences.

Zoho: Zoho[3] is another online workspace which provides some useful tools: word processing, spreadsheets, database reports, presentations, project management, consumer relationship management, application development, wiki, planner, notes taking, chat, mail and meeting room. Zoho offers many services and can be more efficient than Google docs for more evolved collaborative work.

ThinkFree: ThinkFree[4] is an alternative online workspace proposing some classical tools to create cooperative documents such as text document, spreadsheets, presentation and notes. It differs from others by the fact that it is a java-based application embedded in a web-page, whereas most of the others are Ajax-based. This difference allows it to have some extra features and to be the closest from offline applications. Besides, the suite has an interesting aspect which is the possibility to install an offline client in order to keep on working if you lose your network connection. ThinkFree is doubtlessly the most pleasant as it is really close from Microsoft Office[5] applications. However, for the moment, it only proposes four kinds of documents and is not suitable for complex projects.

All these web editors are efficient, in the sense that they allow you to easily create and share your documents with your co-workers. Nevertheless, there is two points that we think are critical: the security and the accessibility of data. Indeed, when you use that kind of service, all your work relies on a specific website that you are not able to manage. Moreover, your documents are stored on a distant server which does not belong to you. In that case, even if you trust the owner of the server you can't be sure that he will not make a mistake and delete or dispatch a part of your work. Besides, if the server goes offline because of a network failure you can't access your documents, or at least you can't share them. For these reasons we think that web-based applications are not yet suited for workers that have to manipulate confidential documents (and in real life, almost all companies documents are confidential.).

1.2.2 Workflow

Workflow is the automation of a business process, in its whole or part. It is concerned with the automation of procedures where documents, information or tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal. Workflow software help improve the process of performing multi-person tasks in the workplace. Some examples of improvement include reduced lag time because manual task routing is eliminated; and better feedback about the state of the tasks that comprise the business process. The first available workflow management systems were developed in the beginning 1980s. Only few of this first generation systems are left in the market. In the next section we review some successful workflow systems.

---

[2] SubEthaEdit : http://www.codingmonkeys.de/subethaedit/.
[3] Zoho : http://www.zoho.com/
[4] ThinkFree : http://www.thinkfree.com/
[5] http://www.microsoft.com/france/office/2007/default.mspx

Lotus Notes[6] is a groupware based on the Internet for the asynchronous collaboration of work groups. Its first version has been released in 1982. Lotus Notes uses a unique database type that allows the combination of structured data (relational) and unstructured (documents, texts…). By linking mails to this kind of database, Lotus Notes allows the creation of communication channels and documents sharing that can be dedicated to some specific functions or work groups. Lotus Notes is composed of:

- A discussion platform with a messaging service and discussion groups;
- A database generator that manages the file and data organization;
- A document generator and indexer which give them properties and access levels.

This division eases the organization, the sharing, the storage and modifications that can be done by several users. Documents are classified in hierarchical sections. Moreover, Lotus Notes allows and maintain the creation of talk threads by ordering request and responses and by linking them to the reference document of the thread. Any search get back to the last state of the discussion and guarantee that documents from any discussion are updated before to get accessed.

Websphere MQ workflow[7] is a workflow management system that is part of the Websphere MQ product line of IBM. It provides two separate tools: a build time environment for managing workflow models, and a runtime environment for managing process instances. Both operate on separate databases; after a workflow has been modeled in the build-time, it has to be exported from the build-time database, and imported into the runtime database. The runtime environment can be accessed using a local Java client that connects to the MQ Workflow server using the Websphere MQ message queuing middleware, or using a browser that connects to a web-interface.

Many other different workflow management systems have been developed like Microsoft Exchange Server[8] (MSE) which is a system capable of providing messaging and a centralized repository of documents and resources for an enterprise. Many companies use Microsoft Exchange to give their employees messaging, calendaring, contact and task management, discussion groups, and document-centred workflow services. Novell Group Wise[9] offer e-mail, calendaring, instant messaging and document management.

Even if Collaborative Editing is one of the first collaboration actions to come to our mind, it is not always the most important collaborative aspect of our lives. Indeed, the simple fact of meeting can be considered in many cases as the ignition or as milestones of the collaboration. Thus, the correct support of meeting clearly becomes a critical aspect of collaborative software. Let's have a look at how this aspect is treated by the literature.

1.2.3 Meeting Support

Within many organizations, meetings are the primary source of communication **[Ellis et al, 1989]**. The concept of meeting support has existed for some time **[Huber, 1982]**, **[DeSanctis and Stefik, 1985]**. Meetings are essential to structure and coordinate work in organizations; and are meant to be sources of stimulation, support, and solutions.

Meeting Support consists of technological and physical environment additions to a conference room. Technologically, networked computers, whiteboards, shared views, and a Group Decision Support System (GDSS) are the major components of such environment. Most CSCW meeting rooms allocate one networked computer per attendee, and network to

[6] **Lotus Note**: http://www-01.ibm.com/software/lotus/products/notes/

[7] **Websphere MQ workflow** : http://www-01.ibm.com/software/integration/wmqwf/

[8] **Microsoft Exchange Serve**r: http://www.microsoft.com/france/exchange/default.mspx

[9] **Novell Group Wise**: http://www.novell.com/products/groupwise/

other devices in the room. The stand-alone whiteboard, an important focus of attention in the regular conference room, is integrated electronically.

Managers spend a majority of their days in face-to-face meetings **[Panko and Kinney, 1995]** so it is not surprising that meeting support is becoming an important focus of information technology.

Yokota **[Yokota, 2005]** propose a video-based discussion support system for several discussion groups that hold meetings repeatedly at the same meeting room. The system records discussions on a video database with some additional information such as timestamps, participant IDs, annotations and links to other materials for reuse. This system aims to extend capabilities of a normal meeting room without major modification of the usage of the room: it avoids requests to users for explicit system operations as often as possible, to realize ubiquitous computing environments. The main usage of this system is: supporting absentees, reminder, annotation and summarization support: users can summarize the past discussions using annotations and the retrieval function. The system assumes following situations: (1) some discussion groups are in the same meeting room. Different discussions will take place by turns in the room. (2) Each group repeats discussions. (3) All participants have some Bluetooth devices (wireless phones, PDAs) as their IDs.

Hu et al propose POEMS (Paper Offered Environment Management Service) **[Hu et al, 2007]** a modularized meeting service management system which uses a digital pen and paper as its interface, which supports paper buttons, controls based on time-stamped drawings, scribbling, and a new type of interaction. It allows meeting participants to control services in a meeting environment through a digital pen and an environment photo on digital paper. This service is deployed in a corporate meeting room with three large public displays.

Many meetings are not successful due to lack of planning and preparation **[David et al, 2006]**. CSCW literature proposes pre-meeting support systems aiming at increasing meeting productivity. A number of basic issues, implemented in pre-meetings can minimize mistakes as well as optimize resources to be considered in the meeting phase: (1) which participants should take part in the meeting?; (2) what is each participant role?; (3) which techniques and methods should be applied in discussions?; (4) what will be the agenda items and the scheduled meeting time?, and finally; (5) what are the expected results?. SISCO **[Borges et al, 1999]** and PRIME **[Guerrero and Pino, 2002]** systems were implemented to support asynchronous pre-meetings, aimed at preparing participants for the discussion of next subjects. On both systems, contributions posted during the pre-meeting (issues, positions, arguments and remarks) are stored in group memory. Thus, support to awareness about the past is supplied together with the further event presentation in the shared workspace, from the moment the user logs in the system.

As anyone can imagine, meeting are one of the most important and traditional aspect of the collaboration between co-workers. Consequently, it's obvious that any evolved CSCW should handle meetings. However, meeting support raises many issues: how to make the participants equivalently considered; how to properly manage and interpret participants' interaction, etc. Going beyond the simple fact of meeting, numerous collaborative experiences relie on the learning principle. In this perspective, some systems have been developed to support the collaborative learning processes through the use of computers.

1.2.4 Computer Supported Collaborative Learning

According to Dillenbourg **[Dillenbourg, 1999]**, collaborative learning is often defined as a situation in which two or more people learn or attempt to learn something together, and collaboration involves the mutual engagement of participants in a coordinated effort to solve problems together. Collaborative Learning happens when students work together in small heterogeneous groups to achieve a common academic goal, such as the completion of an assignment, a worksheet, or a project. Examples of collaborative learning

are, *peer learning*: in peer learning, groups of people with similar roles and complementary learning needs a particular area help each other with knowledge based on their own experiences, *tutoring*: is the activity of a teacher supervising a group of students learning together and *project-based learning*: is a model of teaching/learning that shift away from the classroom practices of short, isolated, teacher-centred lessons and instead emphasizes learning activities that are long-term, interdisciplinary, student-centred, and integrated with real world issues and practices.

The typical characteristics of collaborative learning are listed as follows **[Roberts, 2004]**:

- *Shared knowledge between teachers and students*: teachers have vital knowledge about content, skills, and instruction, and still provide that information to student. In collaborative classroom, teachers also value and build upon the knowledge, personal experiences, language, strategies, and culture that students bring to the learning situation,
- *Shared authority between teachers and students*: teachers encourage students' use of their own knowledge, ensure that students share their knowledge and their learning strategies, treat each other respectfully, and focus on high levels of understanding;
- *Teachers as mediators*: teachers as mediators adjust the level of information and support so as to maximise the ability to take responsibility for learning, since successful mediation helps students connect new information to their experiences and to learning in other areas, helps students discover what to do when they are stumped, and helps them learn how to learn;
- *Heterogeneous groupings of students*: heterogeneous groupings of students enrich learning in classroom, since the perspectives, experiences, and backgrounds of all students are important for enriching learning in the classroom

Using the definition that Margaret M. McManus (1997) gave of it:

CSCL = CSCW + CL (CL: Collaborative Learning) **[McManus, 1997]**

Lipponen defined that "CSCL is focused on how collaborative learning supported by technology can enhance peer interaction and work in groups, and how collaboration and technology facilitate sharing and distributing of knowledge and expertise among community members" **[Lipponen, 2002]**. CSCL has been getting more and more attention from researchers and occupy an entire sub-discipline within CSCW. Any application that facilitates both cooperation and learning falls under the CSCL umbrella. Some important areas of research include distance learning, teaching rooms, knowledge construction, and shared reality.

Distance Learning is playing an increasingly important role at the college level. A distance learning student is usually a full-time professional taking part-time class. Most courses are viewed as lectures broadcast live (or tape delayed). Interaction with the lecturer and on-campus class is limited to the telephone, and asynchronous text exchanges (e-mail, newsgroups, or the web).

Teaching Rooms are classrooms that incorporate computing technology to facilitate synchronous, face-to-face cooperative learning. Each student usually has access to a networked connected computer whose display can be recessed to give the student a line of sight to the lecturer. The lecturer may have the ability to display information on his computer on a large screen visible to the entire class.

Knowledge Construction*:* Knowledge Construction focuses on collective building of domain understanding. A newsgroup is a basic form of group knowledge construction.

Shared Reality*:* Shared Reality refers to computer constructed worlds where students can explore, collaborate, and learn.

CSCL applications include a wide variety of applications. In this section we list some of the CSCL systems and a short description of each.

MEMOLAB is a learning environment that illustrates the distribution of roles among several agents. The learner solves problems interacting with an expert. The system provides assistance using another agent (tutor) which also monitors the interaction. The distribution of roles is conceived in such a way that the agents are independent of the teaching domain and hence can be reused to build other learning environments **[Dillenbourg et al, 1994]**.

Wang and Johnson **[Wang and Johnson, 1994]** have developed a Collaborative Learning And Research Environment CLARE which is meant to support the collaborative construction of knowledge from research papers. CLARE consists of two different tools: (1) a knowledge representation language called RESRA that serves as a meta-cognitive framework for understanding scientific research literature and the learner's perspectives. (2) A process model called SECAI that prescribes a systematic procedure to guide learners in interpreting knowledge elements. CLARE integrates RESRA and SECAI into a consistent, hypertext-based interface.

Marques et al **[Marques et al, 2006]** also propose a distributed and decentralized infrastructure which has the aim of supporting distributed group learning and team work activities. This infrastructure is based on event distribution mechanisms providing awareness so that participants can be notified and thus be made aware of the progress of the groups they belong to. It describes how to collect and propagate events so as to notify group participants about the activities of others. However, it does not describe how this information should be presented to participants and thus some of the effectiveness in providing awareness is lost.

CSCL offers promising innovations and tools for restructuring teaching-learning processes to prepare students for the emerging knowledge society. The success of CSCL applications depends for a great extent on the capability of such applications to embed information and knowledge from the group activity's interaction and use it to achieve a more effective evaluation of collaborative learning **[Avouris et al, 2003]**.

Although most of CSCL systems support many aspects of the CSCL domain, they do not entirely contemplate the users' fundamental needs for collaborative learning environments, such as dynamic support to group awareness, specific components for awareness management, and interoperability between different applications to support collaborative work. Finally, no analysis of the real students' skills and intentions are provided from the rationale of a specific conceptual model for data analysis and management and as result the awareness and feedback capabilities are strongly restricted.

1.3. Synthesis

As we have seen, collaborative systems have evolved according to different dimensions. Earliest systems were only designed to support classical single user applications with a more a less effective collaborative part. Indeed, first systems were collaborative writing software with no other possibility than to edit files one after another. As almost anyone knows, computer science and information technology are fast evolving domains. New methods, principles, ideas and subdomains are emerging every year. CSCW has not been spared by this evolution and basic systems are becoming more and more complex.

Furthermore, CSCW systems have evolved in their use. They are not reserved to scientists, IT professionals or some companies. Nowadays anyone can use a collaborative system in order to share private documents. In addition, more and more companies are using collaborative software to organize their every day's tasks and collaborate efficiently. Indeed collaborative systems allow companies to spread their teams on different locations without losing contact or breaking workflow.

However, for the time being the different tools that have been developed are rarely compatible or lack of the communication that could ensure a simpler collaboration of humans. Hence, one of the most classical issues encountered in collaborative situation is the multiplicity of collaborative system. For instance, if an IT service provider is hired by a client to perform some specific task and developments, the service provider will surely have to use the collaborative tool of the client while it already has one for internal collaboration. Thus, the use of two different collaborative systems will surely force him to perform extra tasks (such as replicating events from one calendar application to the other). Our thought concerning this precise point is fairly clear: developing another kind of environment to support the collaboration without thinking to the compatibility with other applications would be of no use. In this perspective, we think that it is almost fundamental to propose a way to make already existing applications, services and systems cooperate.

The collaborative aspects have evolved, as we will see in models discussion, there were many attempts to model the collaboration through the use of different perspectives.

Chapter 2 Modelling CSCW

The multiplication of mobile devices such as phone, smart-phones, PDA, pocket-pc, netbooks, tablets and laptops implies that users may not have the same resources for collaboration, requiring CSCW systems designers to find new way of interactions allowing users with various, or varying, resources to collaborate seamlessly.

The concept of collaboration is complex, even for humans it is not always simple to see all involvements of collaboration between users, especially when the number of people and the complexity of their organisation raise. Thus if you want to design a collaborative system, it is vital to rely on a robust model. In this chapter we will present and discuss some of the most interesting and promising collaboration models. Starting from models depicting the collaboration as the intersection between communication, coordination and cooperation, we continue by presenting system models, task models and roles models to finally reach group awareness models.

2.1 Collaborations Models for CSCW

The main purpose of CSCW systems is to handle the collaboration between users. To do it they can rely on technical evolutions and tools adaptation for multiple users. But collaboration raises problems that are going far beyond technical issues. Indeed, the main problem of CSCW is the collaboration itself, how a system can effectively support collaboration patterns and how it can be aware of the current collaboration status. To tackle these problems several models of collaboration have been proposed. Among them, some kinds sounded more promising: tasks models, roles models and the refined collaborative awareness models. In the following we will illustrate these models by presenting some of their related researches.

2.1.1. 3C model

3C Collaboration model was originally proposed by **[Ellis et al, 1991]**. It is meant to support the domain analysis phase of the groupware engineering development cycle, by leading designers to explore group collaboration from three related aspects, namely communication, coordination and cooperation. This model has several variations; we can cite the work done by **[Laurillau and Nigay, 2002]**, they define 3 classes of functionalities named communication, coordination and production. It is similar to the 3C model in terms of functional specification of collaborative systems, insofar as both deal with the 3 classes of functionalities that must have computational support in groupware.

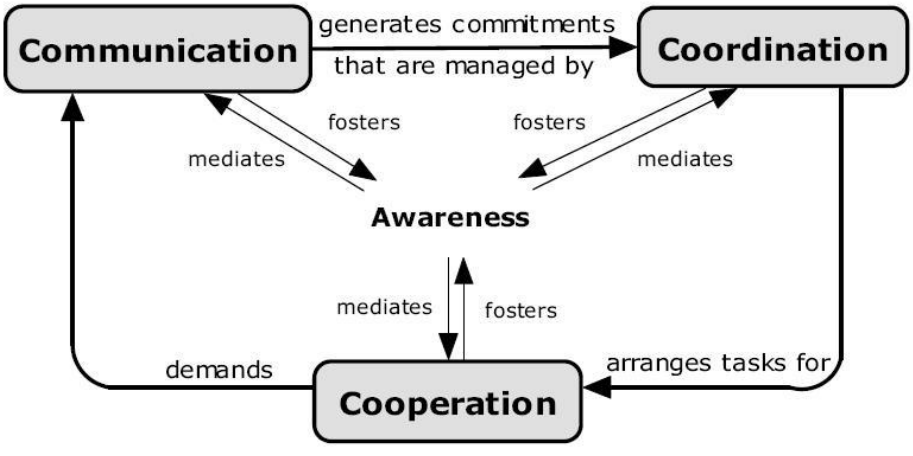

Figure 1 3C collaboration model [Gerosa et al, 2006]

Gerosa et al **[Gerosa et al, 2006]** propose another variation of the 3C model, their model is based on the idea that to collaborate, members of group communicate, coordinate and cooperate as shown in Figure 1.

Communication in group work involves the negotiation of compromises and knowledge. Through coordination the group deals with conflicts and organizes itself so to avoid that communication and cooperation efforts are lost. Cooperation is the joint operation of the members of a group in a shared space. Through perception an individual obtains feedback on one's actions and feed through of the actions of its. In this work, negotiation is seen from a social perspective. Therefore, the process can be defined as a form of group work in which allied negotiators share information to seek a better agreement. In a negotiation, even if each negotiator has autonomy in relation to decision-making, where each one can use a different strategy and hold one's own interests, even then the collaboration between allied negotiators can aid in the search for good results.

Another adaptation of the 3C Model was proposed by Chang et al **[Chang et al, 2006]** to facilitate an effective and efficient human cooperation. As shown in Figure 2, the 3C model represents a three-layer supportive mechanism, namely: communication, coordination and collaboration.

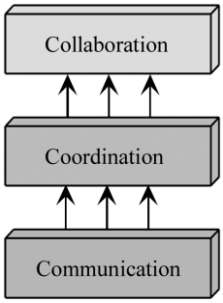

Figure 2 Three layers in CSCW

Communication layer: provides fundamental communication channels. It is not always feasible or even desirable for all communication among collaborators to be face-to-face interactions. However, it is necessary that the communication medium and methodology replacing face-to-face interactions strengthen the sense of group, without detracting the group from collaborative tasks **[Cohen, 1996]**.

Coordination layer: refers to the methods of helping people to work together harmoniously and managing interdependencies among processes, people and available resources **[Malone and Crowston, 1990]**. Such a coordination requires a set of rules, either formal or informal, to regulate the conduct of the participants [**Nicollin and Sifakis, 1994]**.

Collaboration layer: is the ultimate goal that requires a space to facilitate collaborators to share common and ever evolving information among them. Within such a collaboration space, the group must be aware of what others are doing with respect to the collaborative task **[Ellis et al, 1991]**, which is often referred to as "group awareness".

The comparison between the approaches of **[Chang et al, 2006]** and **[Gerosa et al, 2006]** shows a clear difference in the consideration of awareness. Indeed, while Chang considers that the awareness only comes at the highest level (i.e.: collaboration), Gerosa suggests that the awareness in the collaboration has to be dealt at every level of the model, including the fact that each layer can contribute to awareness mechanisms. For us, despite the fact that Gerosa's approach can lead to more complexity, it seems obvious that it can provide a more effective basis to build awareness mechanisms.

2.1.2. System model

Duque et al **[Duque et al, 2008]** propose an interesting approach to groupware model. In their work they investigate different aspects of groupware systems: the application domain, the communication between users of a working group, policies for turn-taking in the use of a shared workspace, definition of tasks and awareness mechanisms. During their work they developed a tool named SPACE-DESIGN which is a groupware system supporting synchronous distributed collaborative work and allowing users to build a design in any kind of domain. They developed this system in order to be domain-independent, to do it they assume that the domain have to be defined in an external repository that the system can inspect to reconfigure itself. They pointed out the fact that most of the systems that claim to be domain-independent have a great lack of flexibility. They also noticed that domain-specific systems possess greater awareness, communication and coordination abilities than domain-independent. They conclude that domain-independence is acquired at the cost of basic functionalities.

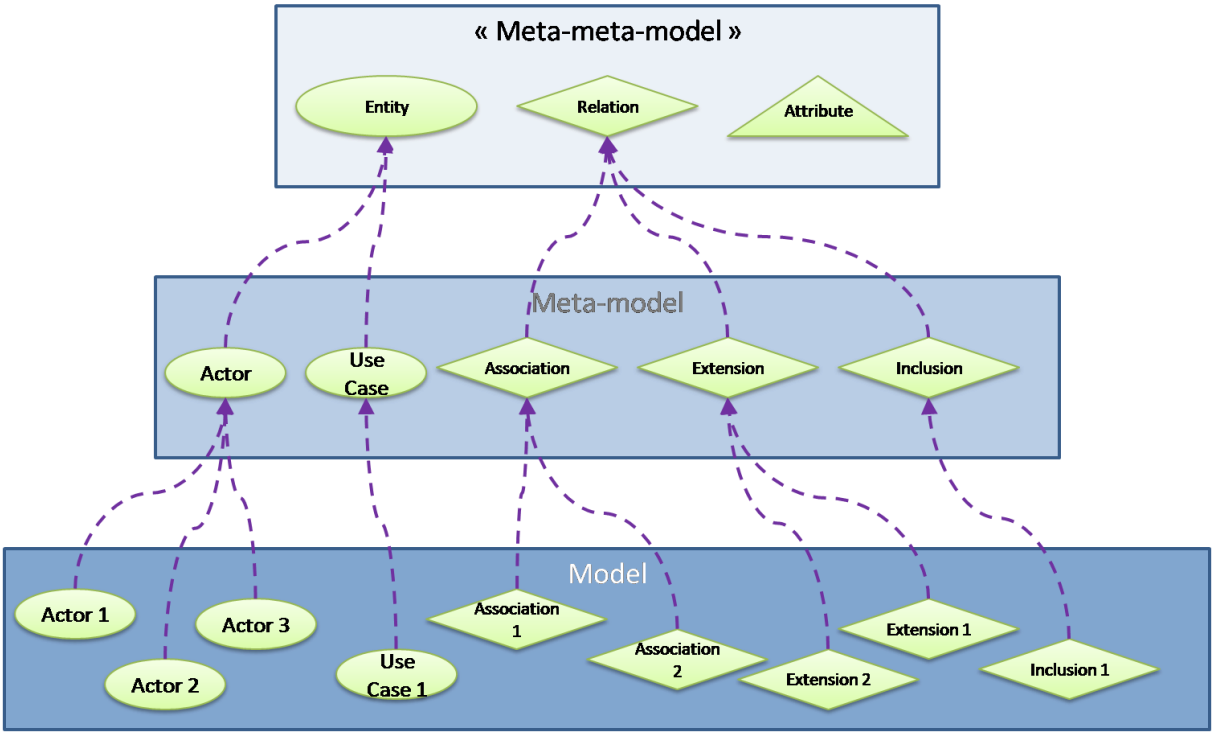

Figure 3 Duque's domain model layers

In order to tackle those problems, they propose to use three levels of representation. The Figure 3 is an illustration of the different levels they use. The first one (at the top of the figure) is called the "Meta-meta-model". It defines the three high-level concepts:

- Entities: this concept is aimed to represent the main concepts of a domain;
- Relationships: they are the links between entities, they describe how entities are related which each other;
- Attributes: they are characteristics of entities and relations between them.

The second level is the "Meta-model". It represents the different elements of a specific domain and the relations between those elements. On the figure below we took the example that Duque explains in his article: the Use Case diagram. The resulting model for this domain contains entities like "Actor" and "Use Case"; it also contains relations like "Association", "Extension"...

Finally the last level named "Model" results from the use of the meta-model in a specific case. This level is the one to be produced by the users of the system.

This domain model looks promising, but if you look a little closer you quickly figure out that it is not so revolutionary. Indeed, the "Meta-meta-model" that is one of the keys to build a real domain-independent application is really classical; it could be seen as a redefinition of the basis of a language such as OWL[10]. The meta-model is interesting as it is the representation of a domain. However we must point out the fact that this meta-model has to be produced by some expert to be complete and correct.

We'd like to point out a specific fact not really explained in the article: as for now it doesn't seem possible in this system to allow users to dynamically add, modify or remove entities from the meta-model. We assume that it is possible to change the XML file representing the meta-model and restart the application with the new meta-model. But we think that, as their application is designed to allow users to produce designs it could be interesting for them to dynamically build or modify the meta-model on which they base their work.

To define domains, SPACE-DESIGN use a set of three XML files: the domain file, the task file and the communication file. The domain file is the definition of the meta-model. The task file is used to represent the goals to be reached by the tasks. To do it they describe goals by definition of constraints and requirements. This constraints and requirements refer to the elements defined in the domain file. The last file, the communication one allow the system to support the structured chat communication mechanism **[Gallardo et al, 2007]**. This chat naturally proposes to the user the list of elements defined in the model to facilitate the communication between users about a specific item.

To solve coordination issues, SPACE-DESIGN proposes three kinds of collaboration which are specified in the task file:

- *Simultaneous work*: any user can work on the shared-workspace at any moment. Blocking is used in order to avoid inconsistent states.
- *Turn assigned by agreement*: the access to the workspace is acquired by submitting a request to the vote of all users.
- *Turn assigned by request order*: this mode works on the FIFO policy: the first to send a request to use the workspace will be the first to use it.

In addition, SPACE-DESIGN incorporates a set of tools to facilitate the work of users (electronic whiteboard, session panel, options toolbar, drawing toolbar …).

Despite the fact that we can pick up some limitations for this system we still admit that this is a great work and that it is good base to build a full system. We think that the meta-meta-model approach is a good idea in the sense that it is a key element to be domain-independent, but we also think that this representation is a little too simple, not really the meta-meta-model itself, but rather the fact that the domain is directly build on it. We suggest that there may be some kind of intermediate level in the representation of the model, a level between the "meta-meta" and the "meta". The authors have led an evaluation campaign of their system. This evaluation pointed out it still has weakness in term of entities representation and task representation mainly. Despite these weaknesses, the evaluation showed that the system is appreciated by users.

On a different consideration of the collaborative work, **[David et al., 2007]** proposes an original work aimed at facilitating the modelling and the understanding of collaborative scenarios. The ORCHESTRA formalism provides a simple way to graphically represent collaborative activities. The initial objective of this work was set to allow a better expression

[10] OWL : Ontology Web Language, http://www.w3.org/TR/owl-features/

of the Cooperative Behaviour Model (CBM) which was designed to represent cooperative scenarios. This model is based on the use of five elements: *actor* which represent the entity (human or group of humans playing a single or a group of roles) performing an action; *activity* which corresponds to the action performed by the actor, these activities are divided into *acting*, *observing* and *editing* ones; *artefact* is a tools or an object used to perform the activity; finally *context* is defined as a set information relative to three aspects: platform, location (logical or physical) and user preferences.

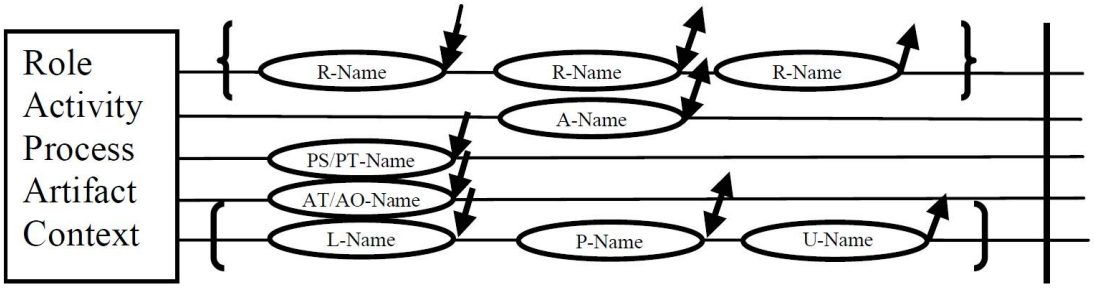

Figure 4 ORCHESTRA formalism

On Figure 4 is summed up the Orchestra formalism. As it can be seen, the author has based their representation on the use of a staff with five chords corresponding to the elements of CBM. In addition, different icons are used to represent information of the scenario:

- {…} parenthesis are used for mandatory situations, (…) parenthesis are used for alternative situations, […] parenthesis are used for optional situations;
- @ is used to represent asynchronous coordination without answer delay, @@ is used for asynchronous coordination with a defined delay;
- & is used for synchronous "in-meeting" cooperation while && is used for synchronous "in-depth" cooperation;
- ● is used for short term (implicit) collaborations and ▬ is used for long term (explicit) collaborations;
- One-way incoming arrows are used for 'acting' activities, one-way outgoing arrows for 'observing' ones and two-way arrows for 'editing' ones;
-  indicates no collaboration awareness,  indicates awareness for some of the actors and  means an overall awareness for all actors.

This work is particularly interesting as it provides a simple and more natural way to represent collaborative scenarios that can be quickly and efficiently apprehended. Indeed, the representation of collaboration scenarios can be extremely complex in some cases and such work can be highly profitable for people dealing with these scenarios.

2.1.3. Task model

In this section we examine existing approaches in collaborative task model and review commonly used formalisms and relevant related work. Task modelling is a well-established research field in Human Computer Interaction (HCI). Task modelling is the art of capturing the tasks and their temporal relationships and dependencies on one another. A task is an activity that should be performed in order to reach a goal. A goal is either a desired modification of state or an inquiry to obtain information on the current state **[Mori et al, 2002]**. Task models' goal is to identify useful abstractions highlighting the main aspects that should be considered when designing interactive systems. The main advantage of task models is that they represent the logical activities that an application must support in order to better

understand the user, his work and his expectations; there are two types of task models: descriptive and prescriptive. The descriptive task models describe the way in which the task is performed currently (even in a manual way, or supported by another system). The prescriptive task model describes how the task should be supported by a new developed system **[Tarta, 2004]**.

The task model is now widely accepted by the CSCW community to be one of the bases to represent collaborative work. It has been the subject of many articles and is still an active research field. ConcurTaskTrees (CTT) **[Paterno, 1999]** has been extended to CCTT (Collaborative ConcurTaskTrees) **[Mori et al, 2002]** in order to support the specification of collaborative (multi-user) interactive systems. A CCTT specification consists of multiple task trees. One task tree for each involved user role and one task tree that acts as a "coordinator" and specifies the collaboration and global interaction between involved user roles. Main shortcoming of CCTT is that the language does not provide means to model several actors, who simultaneously fulfil the same role. Klug and Kangasharju **[Klug and Kangasharju, 2005]** as well as Bomsdorf **[Bomsdorf, 2007]** introduced an extension to CCTT where a task is not regarded as an atomic entity (like in CTT) but has a complex lifecycle, modelled by a so-called task state machine. The former approach does not consider a temporal operator as state chart whereas the latter does not consider abortion or skipping of tasks.

Groupware Task Analysis (GTA) **[van Welie et al, 1998a]**, **[van Welie et al, 2003]** is another example of such task-based approaches that combines aspects from several methods. GTA is a conceptual framework for groupware task analysis that aims at helping designers' analyses the users' task world from three different viewpoints, namely agents, work and situation. Each viewpoint draws the designers' attention to specific elements of the task world.

Keeping in mind the necessity to improve the general ergonomics of collaboration, Molina et al **[Molina et al, 2008]** propose a generic methodology for the development of groupware user interfaces. This approach is called CIAM (Collaborative Interactive Applications Methodology); the approach defines several interrelated models capturing roles, tasks, interactions, domain objects and presentation elements. Even though, the models cover all important aspects involved in groupware user interfaces, they are only used at the analysis stage. Subsequent development phases (e.g. requirements or design) are not covered. These works fail to account for user roles and multiple role-based views on the same collaborative task. Aiming at fulfilling this gap Vellis **[Vellis, 2009]** has adopted an extended version of CCTT, which would be taking care of user role differentiations and their effect in the whole process.

With a radically different approach, Van Welie et al **[van Welie et al, 1998b]** have proposed an ontology that captures the most important aspects of the world, it describes task model on a meta-level. The ontology defines the basic concepts and relationships that are relevant for the purpose of task analysis. The main concepts are *Goal*, *Task*, *Agent*, *Role*, and *Event*. Herczeg has also developed an ontology **[Herczeg, 1999]** for task models. He calls his model a "task analysis framework".

As tasks are always performed within a certain context or environment, their interplay with the environment should also be taken into account. This issue was first tackled by Dittmar and Forbrig **[Dittmar and Forbrig, 2003]** who proposed modelling the execution environment in accordance with the task specification. The environment captures the domain entities which are manipulated, created or needed for the performance of a certain task. Unfortunately the approach by Dittmar and Forbrig is not very well integrated with standard software engineering models.

In a different perspective Penichet **[Penichet et al. 2007]**, **[Penichet, 2008]** proposes a task model for CSCW based on the use of several well-known task modelling aspects. Their model is aimed at describing "the tasks that should be performed to achieve the application goals" by giving them a good characterization. This model is aimed at designers that have to design groupware systems. What they propose is not a complete new model of tasks but a

new “composition” of existing tasks model in order to have a better, more complete and more effective task model. Their approach is based on the description of tasks that are realised in groupware systems keeping in mind more classical aspects and mechanisms to analyse them. They argue that classical CSCW features or time-space features are not enough to correctly describe a groupware, but that a well done combination of them can do it.

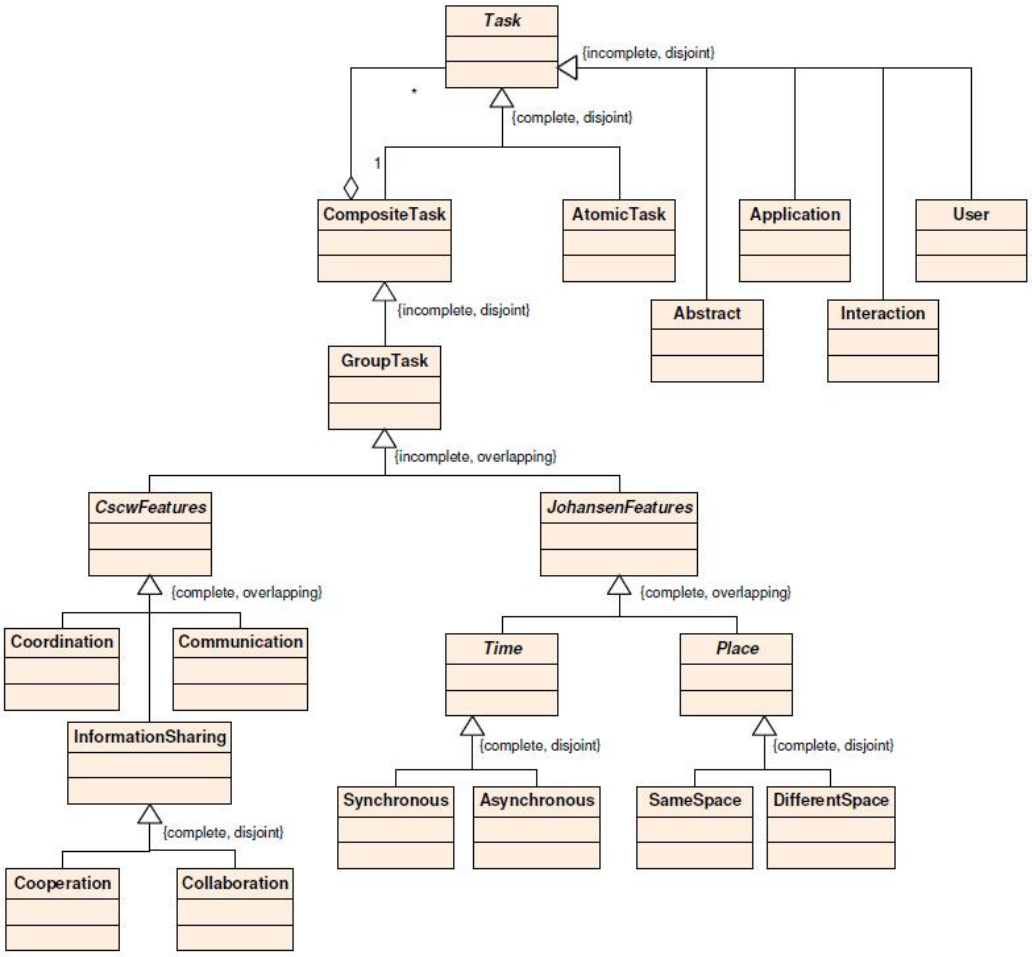

Figure 5 Penichet's Task Model

The Figure 5 shows how the authors of this article have designed their model. On this figure they do not represent the full model they use to describe a task, but only the part that is needed to characterize it, so we need to take into account that a great part of the task information is missing as they are not characterizing (Users, Roles ...). On this figure we can make a first difference between composite and atomic tasks. Atomic tasks are tasks that can’t be divided into others. On the contrary, composite tasks are combination of several tasks that has been assembled into a global one. Their model also relies on the classification provided by **[Paternò 1999]** which allows identifying tasks to be performed in order to accomplish a specific goal. Groups' tasks are defined in term of CSCW features and Time-Space features. CSCW features are *coordination*, *communication* and *information sharing*. *Information Sharing* is refined in term of *Cooperation* and *Collaboration*. They based this description according to **[Greif and al. 1988]**, **[Grudin, 1994]**, **[Poltrock and al. 1999]** and **[Poltrock and al. 2005]**. Time-Space features are based on the one given by **[Johansen, 1988]**. It has to be noticed that authors proposed an example based on a whiteboard application where they are able to make a good description of the tasks involved.

This work is interesting as they base their model on existing and accepted ones. Moreover they propose a model able to facilitate the task of groupware designers. It is also valuable as they don't provide a rooted model but a good base to build on.

Even though, the models cover all important aspects involved in groupware user interfaces, they are only used at the analysis stage. Subsequent development phases (e.g. requirements or design) are not covered. The methodology is not assisted by a tool which would facilitate the creation and simulation of the various models. In particular, the latter is an important shortcoming since the animation of models is an important technique to obtain stakeholder's feedbacks.

Task models are interesting, because they can be easily understood by humans as they represent "classical" organization of collaborative work. But some models take different ways to represent the collaboration, making them interesting by the simple fact that they have new points of view of the collaboration.

2.1.4. Roles model

For numerous researches, the most interesting aspect of the collaboration isn't task model. Indeed, some of them have focused their interest on the fine description of users inside groups and teams. Consequently, these researches have produced different kind of model centred around the roles played by humans.

Role-based Collaboration (RBC) is a methodology to design and implement new computer-based tools. It is an approach that can be used to integrate the theory of roles into CSCW systems and other computer-based systems. It consists in a set of concepts, principles, mechanisms and methods **[Zhu and Alkins, 2007]**. RBC is intended to provide some benefits to long-term collaboration: identifying the human user "self", avoiding interruption and conflicts, enforcing independency by hiding people under roles, encouraging people to contribute more and removing ambiguities to overcome expectation conflicts. It is also intended to provide benefits to short-term collaboration: working with personalized user interfaces, concentrating on a job and decreasing possibilities of conflicts of shared resources, improving people's satisfaction with more peoples' playing the same role during a period and transferring roles with requirement of a group. Finally, in management and administration, it helps at decreasing the knowledge space of searching, creating dynamics for components and regulating ways of collaboration among parents.

Some CSCW systems have indeed applied the concept of roles. Barbuceanu et al **[Barbuceanu et al, 1998]** have proposed role based approaches to agent coordination. This approach includes a "practical, implemented coordination language for multi-agent system development" that defines agents, their organization and roles. Agents play roles in an organization, and a role is defined by its major function, permissions, obligations, and interdictions. A role's permissions include agents under its authority and its acquaintances. An agent's beliefs and reasoning are partitioned on the basis of the roles it plays to facilitate context switching **[Barbuceanu et al, 1998]**. A combination of events leads to a situation for the organization, with each agent member in a given local state. An agent's behaviour in a situation is determined by its conversation plans, and these are usually specified to be between a particular pair of roles.

Edwards **[Edwards, 1996]** propose a system that can implement a variety of useful policies in collaborative settings, particularly in the areas of awareness and coordination. This system uses the notion of roles to associate categories of users with particular policies. Intermezzo roles can represent not only groups of users, but also descriptions of users in the form of predicates evaluated at runtime to determine group membership. Dynamic roles, in particular, expand on one of the central themes in this work: by bringing information about users and their environments into the system, it can make computer augmented collaboration more responsive, and can free users of many of the implicit burdens in working with today's collaborative systems.

In a more recent article, Zhu **[Zhu, 2004]** proposes his view of collaborative authoring based on the use of roles. He points out the fact that collaborative systems should not only support virtual face-to-face collaboration between distant people, but should also improve physical face-to-face by providing mechanisms to overcome drawbacks of face-to-face collaboration. They notice that WYSINWIS (What You See Is Not What I See) can be an efficient model for the development of collaborative systems. Thus WYSINWIS systems can allow different users to have different views of a shared document according to their roles inside the collaboration. This kind of interaction is not totally new, and such systems exist for a long time, but what they propose is a mechanism based on the precise role definition and specification to allow roles to be dynamically tuned and managed in the system.

Furthermore, Zhu and Tang **[Zhu and Tang, 2007]** propose a role based hierarchical group awareness model (RHGAM). Firstly RHGAM constructs a group cooperation environment (GCE), which is then extended by group awareness content, awareness hierarchy, the task decomposition rule. The model divides the awareness information into four levels by decomposition and recombination using a role-task graph and the thinking of group structure. In RHGAM, role is the basic of group cooperation; with the different group structure and task relation, the awareness information is shared between roles hierarchically.

In a relatively different perspective, Ahn et al **[Ahn et al, 2003]** implemented a role-based delegation framework to manage information sharing (FRDIS) for collaborating organizations. Their central idea is to use delegations as a means to propagate access to protected resources by trusted users.

Role models propose a "natural" approach to collaboration; with the help of task models it is possible to have an accurate description of user's collaborative work. Roles can seem simple, but describing them correctly with all their characteristics is a really complex issue.

2.1.5. Group awareness

In CSCW, group awareness is the pre-condition of collaborative work. This concept defines the ability of a user (or a system) to stay aware of the current state of the group implied in the collaboration. Good group awareness model can facilitate the research of group awareness theory, reduce the cost and conflicts, and increase the efficiency of collaborative works **[Gao-feng et al, 2006]**. There are several kinds of group awareness models, one of the first of those is the spatial model of interaction in large virtual environments proposed by Benford and Fahlen **[Benford and Fahlen, 1993]** this model aims to provide flexible and natural support for managing conversations among large groups gathered in virtual space. However, it can also be used to control more general interactions among other kinds of objects inhabiting such spaces, they depict the awareness intensity between two actors by the intersection and union operation of the objects in users' interest space and effect space, the notion of awareness is used as the basis for controlling interaction and the model provides mechanisms for calculating awareness levels from the spatial properties of objects (e.g. position and orientation). In short this spatial model defines key concepts (Aura, Focus and Nimbus) for allowing objects to establish and subsequently control interactions. Aura is used to establish the potential for interaction across a given medium. Focus and nimbus are then used to negotiate the mutual and possibly non-symmetrical levels of awareness between two objects which in turn drives the behaviour of the interactions. Finally, adapter objects can be used to further influence aura, focus and nimbus and so add a degree of extendibility to the model. This model is very interesting and forms an important aspect of support for CSCW, but wasn't well-combined with collaborative mechanism. However, another benefit of such work is that by quantifying spatial interactions across media and by providing a way to measure awareness (even if these measurements have to be interpreted), it forms a valuable model and an interesting base to build a statistical model and feed any associated statistical framework. This kind of opportunity can then provide relevant information about the

collaboration in order to improve its mechanisms, but it can also provide interesting feedbacks on the model used to support the collaboration.

Rodden **[Rodden, 1996]** extended the spatial objects awareness model to depict the relations among collaborative applications in non-share work domain. He measured the awareness intensity by information flow chart among application. These two models success in describing group awareness characteristics, but they do not really include group structure into their measure.

Daneshgar and Ray **[Daneshgar and Ray, 2000]** propose a hierarchy awareness model which can be summarised as below:

- Identify various human roles, their tasks, and interactions involved in a cooperative management environment.
- Define awareness levels required from each role when performing a task.
- Identify awareness levels actually supported for the above role-task tuples.
- Identify the gaps in awareness levels supported in various role-task tuples.
- Design tools and artefacts to plug the gaps identified.

This model is simply using awareness hierarchy to measure the cooperative level of different actors in collaborations.

All these awareness models mentioned above have a disadvantage in common, all of them could only characterize the awareness intensity among actors in a coarse scale, and none of them can measure it by precise mathematical calculations.

In an analytic approach Yan and Zeng **[Yan and Zeng, 2004]**, **[Yan, 2009]** propose an original model for group awareness inside CSCW systems. They assume that there are mainly two aspects in group awareness: "group awareness model" and the "method of realization". They point out the fact that for now, the main problem is still the description of group awareness characteristics and form. What they want to do to solve the unburied problems is to analyse basic elements of group work: "task", "action" and "role". One of the main aspects to consider in their approach is that the authors assume that "*task has a thinner granularity in describing the group awareness*". They demonstrate this by making the assumption that if the role changes, so does the task. But if the task changes, the role does not necessarily change. Moreover, a change on the action may change the task but does not alter the associated role. It is why they put the task on the foundation of their awareness model. The proposed "Task-Based Group Awareness Model" is subdivided in two parts: information management and information forming. Information Forming part includes three modules:

- Task disassembling module: has in charge to divide a task into sub-tasks until it reaches "atomic tasks" which are undividable tasks. The module produces a task tree representing the disassembling of tasks.
- Awareness intensity module: gives an awareness value to each atomic-task. And so, recursively give a value to each node of the task tree.
- Task assembling module: assembles the awareness values of task according to some assembling rules. This module provides awareness information to be distributed.
- The Information Management part is also divided into three modules:
- Information collection module: feeds the whole awareness model with information. This module incorporates information filters in order not to be flooded by too much information.
- Information distribution module: manage the routing of information. That is to say, it decides which information to send to which user.
- Information showing module: handles the display form of the information.

The more interesting module of this application is obviously the task disassembling one, without which nothing could be done. To understand how it does we have to introduce some of the theory behind this work. The elementary definition here is the formal definition of a task. A task is defined as a triplet *T:(Role1, Action, Role2)* where T is the task, Role1 and Role2 are roles associated with the task. Role2 is only mandatory when Role1 cannot complete independently the Action. So, to disassemble a task, the system recursively disassemble Role1, Action and Role2 until he can't divide any more. When he reaches this state, the task is defined as "atomic". They also define a set of rules for disassembling in order to avoid inconsistent state. Moreover they point out that task disassembling is time consuming, that's why they propose to pre-process most common tasks categories into task tree templates. The other part that could have been interesting is the awareness intensity value. Unfortunately this value heavily relies on predefined intensity policies.

In [**Yan and Zeng, 2004**], the measurement of awareness intensity was more concerned in a new group-awareness model based on role and task, a measurement based on role difference was proposed. However, this model was based on a static group structure, which cannot represent the dynamic property of group structure; therefore, this model cannot precisely characterize the changing tasks, roles and activities in the real world.

The proposition made by Yan and Zeng is based on an interesting approach of group awareness. The task-based awareness seems to be promising. However we have to point out several lacks. Firstly, this work is mere theory, when they wrote it there was no implementation of their work, not even a prototype, that's why we have to be dubitative concerning the real efficiency of this model. Moreover they reasoned by telling that as task depends on role, task is a more efficient group awareness descriptor. But they also showed that task depends on action. If we follow their argumentation we should say that action is an even more effective descriptor, which they do not mention. Finally, we do not really agree to the task representation they propose. By example we can say that, depending on the group in charge, a task will not be handled the same way, with the same roles and actions, even if the goal is the same. That's why we think that task and roles should not be depicted by some stilling trees and templates but that it should be dynamically extracted and modified during the group collaboration.

2.2 Ontology

In order to correctly model the collaboration of users, one has to use the adequate structure. This kind of representation can be done with trees, graphs or any other type of classical structure. However, a more and more employed concept is the one of Ontology. This really old concept (which can be traced back to the Greeks) refers to the study of very existence of things, how they organized themselves, how they're related, how they can be grouped, which are their similarities and what differentiates them. On a more pragmatic point of view, ontologies are now used in computer sciences to represent how information can be organized. Ontology is defined as an explicit specification of a shared conceptualisation by Gruber **[Gruber, 1993]**. Defining an ontology is a modelling task based on the linguistic expression of knowledge **[Bachimont, 2000]**. In order to do that, ontologies consist in a set of various components:

- Classes: they can be seen as classes in object-oriented programming, they define sets of characteristics shared by instances of this class; ex:
- Attributes: characteristics held by objects and classes;
- Relations: the expression of how objects and classes are related;
- Restrictions: descriptions of constraints over the existence of objects, they express features and characteristics to ensure the coherence and the acceptance;

- Rules: statements in the form of an if-then (antecedent-consequent) sentence that describe the logical inferences that can be drawn from an assertion in a particular form;
- Axioms: assertions, each one describing a part of the theory represented in the ontology. Contrary to classic axioms, they not only describe the initial state of the knowledge but also the resulting statements.
- Events: the changing of attributes or relations;
- Individuals: instances or objects;

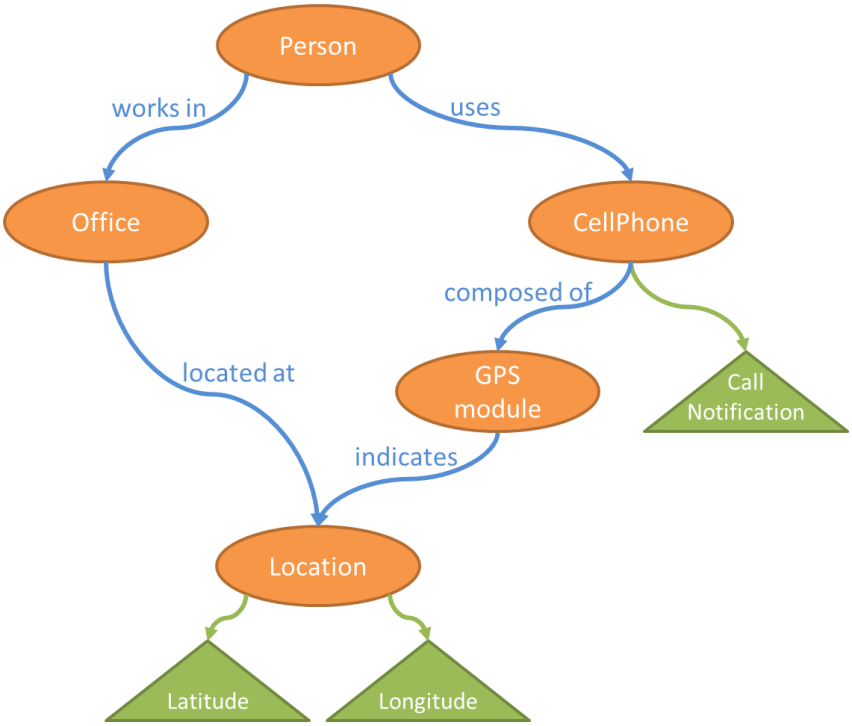

Figure 6 Ontology Example

On the Figure 6 we have depicted a simple example of ontology, representing a limited part of user's context. From the top to the bottom: we have represented a class "Person" which is related to the class "Office" through the "works in" object property (relation) and to the "Cellphone" class through the "uses" one. The "CellPhone" class is related to the "GPS Module" class by the "composed of" object property. In addition, the "CellPhone" has a datatype property (attribute) "Call Notification" which indicates how its owner wants to be notified of a call. Both the "GPS Module" and the "Office" classes are related to the "Location" class which is composed of two attributes: "Latitude" and "Longitude".

In addition to this structure, we can define a quick rule composed of a set of axioms, helping us deduce new facts from the current context:

IF (There is a Person P, and P works in an Office O, and P uses a CellPhone C, and C is composed of a GPS Module G, and O is located at a Location Lo, and G indicates a Location Lg, and Lo has a Latitude LaO, and Lo has a Longitude LgO, and Lg has a Latitude LaG, and Lg has a Longitude LgG, and LaO equals LgO, and LgO equals LgG) THEN (set Call Notification of C to "Vibrate").

In this example, each axiom is separated by a coma, when we put these axioms together they form a rule which can be used to automatically set the call notification of a cellphone to "vibrate" from the moment where its GPS module indicates that it is close to its owner office.

2.3 Synthesis

Modelling collaboration is a domain itself and one of the most complex. Indeed, it refers to the particularly active and discussed domains of ethnology, sociology and psychology. The most popular collaboration models rely on the representation of at least two associated elementary concepts: the role and the task. Tasks are tied to roles in a sense that a specific role implies some specific tasks and that a specific task can only be done by some specific roles. Tasks are unavoidable in the representation of collaboration as work distribution is made by the allocation of tasks. Roles represent users' tasks responsibilities inside the collaboration, in this way they can help characterizing a user. In addition to these two concepts, another issue when you try to model collaboration is the high dynamicity of such relations. Rare are the case where roles and tasks do not evolve during the collaborative activity. This dynamicity is critical to build efficient systems, but often raises problems particularly hard to tackle for producing coherent model.

Social networking is a relatively recent, highly popular, use of computers to allow social related users to keep contact. Social relations concerned by this fashion are various; they can be personal relationships such as friendship, scholar relations, neighbourhood relationship, etc. They can also be professional relationships or else. Social networks websites can help analysts to understand complex social interactions allowing them to design more accurate collaboration models.

Modelling the collaboration between users can't be summed up by task and roles. Indeed, implicit and fast evolving mechanisms appear as soon as two peoples need to communicate. These mechanisms are partially driven by users' personalities, but also by available communication means and by communication needs. What collaboration modelling tries to do is to represent those mechanisms starting from user's observation. We suggest that it is a false problem. We should not start only from users' observation to create models but rather by a combination of how collaboration "should be", how users would like to collaborate and how tools could make them collaborate.

Collaborative awareness model emerges from these different aspects. This new kind of awareness extends the traditional single user context awareness by handling collaboration context. This is done mainly by accessing to some kinds of context information: context of other users concerning their part of work (their tasks if we want to stick with most of models), context of other users that can be relevant for a better collaboration (location, device used, availability, scheduling, expertise...), information of the current collaboration with others (started conversations, shared workspace being used...), information about the collaboration product (the document created by the collaboration, the design developed, the code written) and information about collaboration means (already in use or that could be used to improve this collaboration). The collaborative awareness can then be defined as the ability of CSCW systems to provide relevant collaboration information to the user at the right time. This awareness can also been hidden to the user if the system do not want to overload him. This case appears when the system has to adapt users' own context (for instance an application setting or data) to his "collaboration context". Efforts have been conducted in this way and some platforms and middleware already allows interesting collaborative awareness.

If we put these considerations in perspective of our research we can point out several elements of specific interest: from the system models we presented, besides the need to be domain independent it appears that in order to be really effective, a system needs to be at least partially independent from the collaboration patterns used by the participants. In a second time we think that the use of roles and tasks models is unavoidable for any system wanting to deal with advanced and accurate awareness of the collaboration. The group awareness for its part insists on the fact that any effective collaborative system has to know at least a part of each participant context. By synthetizing those aspects, we came to the conclusion that we needed to build a domain-independent, collaboration-pattern-independent model allowing us to correctly depict users' context and nevertheless find a way to optimize their collaboration.

As it will be presented in the second part of this document, the ontology we built provided us the domain and collaboration pattern independency, while the device collaboration rules (used with our ontology) ensured us of an effective way to optimize users' collaboration by reinforcing and automating devices collaborations.

Chapter 3 Evaluating CSCW

As we have seen until now, the CSCW domain is very active, more and more scientists and companies are working in this field. Consequently there are countless designs, models, frameworks, applications, services and ways to use them. But if there is one specific aspect that is rarely sufficiently considered, it is the one of evaluation. It is even more damaging as it strangles the overall progresses of researches. We can say that, because evaluation should not be considered as a specific part of the researches but rather as a transverse aspect which may impact any level.

This chapter is organized as follows: in a first time we present a deep investigation on the different methods used to evaluate the collaborative work (divided in traditional and original methods). Once this first step is done we can introduce our review of taxonomies of evaluation methods which aim at providing a way to characterize and create classes among those methods. Finally we present some of the numerous evaluation frameworks that have been proposed to help the evaluation of collaborative systems. As we'll see in the next chapters, this survey on CSCW evaluation is critical to properly apprehend our contribution in the domain of evaluation strategy building (which is developed in the seventh chapter).

3.1. Evaluation methods

In computer sciences, evaluation methods are legions, there is almost as many of them as there is of systems. Still, methods allowing the evaluation of Computer Supported Collaborative Work are not so numerous and when we look more closely we quickly come to the idea that only few of them are especially thought to fully support groupware evaluation. Besides, there is no real consensus on what approach is the good to evaluate such systems, leading to the emergence of various methods, sometimes relatively exotic.

In the remainder of this section we explore existing CSCW evaluation methods according to two main categories: traditional methods applied to CSCW and methods especially created for it. Among these two main categories we can point out some similarities: discount methods aimed at providing low-cost evaluation, scenario-based methods and multiple-phased ones.

3.1.1. Traditional Evaluation methods applied to CSCW

CSCW implies complexity, a complexity hard to handle during design phase and development, but even more as soon as we want to evaluate those systems. Given that, it is not a surprise that the first evaluation methods used in CSCW have not been designed especially for it. Indeed, the first tries to evaluate CSCW were done using single user methods. Besides, methods from other domains were employed for CSCW as we thought they could have been more suited. Let's have a look at some of them.

Near the origins of CSCW evaluation we can find the Soft Systems Methodology **[Checkland, 1989]** or SSM. This method is deeply rooted in management and information sciences. The evaluation is conducted without preconceived notions or questions about the nature of the system. Although it is difficult for novices to use SSM, since it is a powerful methodology in the hands of an expert. SSM is still one of the most used methodologies. It is based on seven descriptive stages:

1. Enter situation considered as problematical
2. Express the problem situation
3. Formulate root definitions of relevant systems of purposeful activity
4. Build conceptual models of the systems named in the root definitions
5. Comparing models with real world situations

6. Define possible changes which are both possible and feasible
7. Take action to improve the problem situation

As anyone can see these are the foundations of almost everything we are doing now in term of evaluation. But by being too generic, this method lacks of relevance if we are looking for more specific evaluation methods.

SSM have to be known but cannot really be used as is to conduct a real, full scale evaluation. We'll now present some more advanced methods based on existing ones. In order to progressively introduce more and more complex methods, we begin by the presentation of methods which may not be the most complex and complete, but that ensure users to evaluate their system at a low cost: discount methods.

3.1.1.1. Discount Evaluation methods

One way to address the need for groupware evaluation techniques is to adapt accepted discount methods developed for single-user software usability. Discount methods focus more closely on interface usability issues; they work well with low fidelity prototypes, which allow evaluations to take place during early development when there is no operational prototype for users to test in real work setting. Discount methods for groupware usability evaluation usually fall into three classes **[Ferreira, 2005]**: inspection, inquiry, and analytical modelling. Inspection and Inquiry methods are appropriate for identifying specific problems with collaboration and for providing overall assessments of usability; analytical modelling including engineering modelling, enables evaluators to quantitatively predict human performance with the groupware.

We present techniques that can be done rapidly, such as interface inspection techniques (heuristic evaluation and task-centred walkthroughs), observational methods and subjective assessments by realistic participants. We describe each below:

Groupware Heuristic Evaluation (GHE) **[Baker et al, 2002**] is an adaptation of the Heuristic Evaluation method (HE); HE is a popular discount evaluation method for diagnosing potential usability problems in user interfaces. It defines a particular interface inspection process where several evaluators examine an interface and judge its compliance with some recognized usability principles called heuristics **[Nielsen and Molich, 1990]**. Heuristics draw attention to usability problems often found in single user systems, such as how feedback is provided, how errors are minimized, how help is provided, and so on. Non compliant aspects of the interface are captured as interface bug report, where evaluators describe the problem, its severity, and perhaps even suggestions of how to fix it. HE techniques can be applied to groupware by replacing the current set of heuristics in rephrasing those activities. GHE is based on eight groupware heuristics **[Baker et al, 2001]** which act as a checklist of characteristics any collaborative system should have:

Heuristic 1: Provide the means for intentional and appropriate verbal communication
Heuristic 2: Provide the means for intentional and appropriate gestural communication
Heuristic 3: Provide consequential communication of an individual's embodiment
Heuristic 4: Provide consequential communication of shared artefacts (i.e. artefact feed through)
Heuristic 5: Provide Protection
Heuristic 6: Manage the transitions between tightly and loosely-coupled collaboration
Heuristic 7: Support people with the coordination of their actions
Heuristic 8: Facilitate finding collaborators and establishing contact

Tab 1 Groupware Heuristics

Evaluators who are experts in them examine the interface, recording each problem they encounter, the violated heuristic, a severity rating and optionally, a solution to the problem. The problems are then filtered, classified and consolidated into a list, which is used to improve the application.

This method is interesting, but it is noticeable that it doesn't include users for the evaluation process and that the different heuristics are evaluated by some kind of "experts" who are to be found in a first place, which is not always easy and potentially costly.

On the contrary, Groupware Observational User Testing (GOT) **[Gutwin and Greenberg, 2000]** is centred on users, either on the observation or the interview of them. The method relies on the observational user testing method (OUT). OUT is done by observing how people perform particular tasks on a system in a laboratory setting **[Dumas and Redish, 1993]**. Evaluators either monitor users having problems with a task, or ask users to think aloud about what they are doing to gain insight on their work.

GOT follows the same principle, but focuses on collaboration and analyses users' work through predefined criteria, e.g., the mechanics of collaboration **[Gutwin and Greenberg, 2000]**. This method is interesting as the user is at the centre of the evaluation; however his role is still limited.

Discount usability evaluation methods have been introduced as a way to assess CSCW systems. However, one criticism of these techniques is that they poorly make use of information about users and their work contexts. To address this problem **[Pinelle and Gutwin, 2002]** proposed Groupware Walkthrough (GWA). GWA is an evaluation method based on cognitive walkthrough, a popular evaluation method for single user software **[Polson et al, 1992**]. Walkthroughs is based on the notion of an interface walk-through. There are many variations of how to perform a walkthrough, in all of them the inspector begins with a realistic and detailed task description, a description of the user, and an interface to evaluate. The inspector then 'walks through' the interface step by step by imagining each action the user would take while performing the task on particular system. The GWA method is a substantive modification of cognitive walkthrough. Changes were made to accommodate multiple user descriptions, uncertainly in group task performance, and group work metrics.

For each collaborative scenario:

- Review the scenario to become familiar with users, intended outcome, and surrounding circumstances
- For each task in the scenario:
    - Attempt to carry out each alternate subtask;
    - Record how each subtask was carried out;
    - Record problems, but then assume they are fixed and continue.
- After each task, ask the following questions:
    - Can the task be performed *effectively* - does the interface supply the means to do it (and correctly)?
    - Can it be performed *efficiently* – would the group make the effort required to perform the task?
    - Can it be performed with *satisfaction* – would the group be motivated to do this task, and would they be happy with the outcome?

After all tasks, determine whether the interface allows the group to achieve their overall intended outcome.

Tab 2 Steps in a groupware walkthrough [Pinelle and Gutwin, 2002]

GWA has two components: a group task model: provides a basic framework for analysing contextual information to support usability walkthroughs and a walkthrough process: guides evaluators as they step through tasks and evaluate the groupware interface. In GWA, a scenario is a description of an activity or set of tasks, which includes the users, their

knowledge, the intended outcome, and circumstances surrounding it. In order to construct scenarios, evaluators observe users and identify episodes of collaboration. The process of evaluation in GWA is depicted by Tab 2. Each evaluator, taking the role of all users or one in particular, walks through the tasks in a laboratory setting, recording each problem he encounters. A meeting is then conducted to analyse the results of the evaluation.

This method is interesting as it relies on users' behaviour. Moreover, it proposes a fine evaluation of each task of the collaboration. Even if it looks efficient, we don't think it is sufficient. Indeed, users are part of the evaluation, but their role is a passive one which does not allow them to personally 'evaluate' the system.

Discount methods have advantages; they offer a mean to quickly evaluate a system with limited cost and most of the time with few constraints. Offering so much advantages has a cost, which is paid by limitations, such as dissociation from the real work settings (laboratory experiment are necessary, but cannot fully recreate complex real situations), lack of a real theoretical basis, weak coupling with the domain (being able to evaluate precise aspects of a specific domain with any given method implies a long phase of adaptation or configuration of the method, which is contrary to the discount approach) and lack of accuracy (due to the fact that a discount method is cannot evaluate all aspects of a system and will only evaluate these aspects from a single point of view). In this perspective, more complex and heavy methods have been proposed. The second step of our journey among methods adapted from traditional ones leads us to the group of methods based on scenarios.

3.1.1.2. Scenario Based Evaluation

A scenario is defined as a concrete description of an activity that the user engages in when performing a specific task, a description sufficiently detailed so that the design implications can be inferred and reasoned about **[Carroll, 1995]**. Two groups of scenarios are identified in the literature. The first is the "as-is" scenarios where narrative details are provided on how the operations are currently being performed. These are referred as "problem scenarios" **[Carroll et al, 2006]**, **[Rosson and Carroll, 2003]**. The second group is the "to-be" scenarios which are more visionary and serve as a target for development; they describe how the system could or should work. These are further split into "activity scenarios", "information scenarios" and "interaction scenarios" **[Rosson and Carroll, 2003**].

In computer science a scenario is a description of interactions between users and the system **[Sommerville, 2004**]. They are not intended to be absolute reference of how the system should work but attempt to describe how the system will be used in the context of daily activity **[Armstrong, 2001]**.

Scenarios are typically written in natural language with a minimum of technical detail by usability or marketing specialists who work in conjunction with end users to create realistic scenarios. Tab 3 gives a description of what a generic scenario should be comprised of.

Initial assumption	A description of what the system and users expect when the scenario starts
Normal	A description of the normal flow of events in the scenario
What can go wrong	A description of what can go wrong and how it is handled
Other activities	Information about other activities that might go on at the same time
System state on completion	A description of the system state when the scenario finishes

Tab 3 Simple definition of a scenario [Sommerville, 2004]

Scenarios can be used at different stages of a system development life cycle, although they are better known to be used at the requirements stage. Apart from using scenarios for capturing requirements, there are also attempts to use them in requirements analysis stage. Scenario-based techniques have been suggested as appropriate for system because scenarios

represent concrete instance of system use that can span space, time, people, and system features, while providing designers, developers and other stakeholders with ecologically valid units on which to anchor their analyses. Scenarios may provide a potentially useful alternative to feature set and incident-based evaluation techniques that are more narrowly focused on specific artefacts, attributes or events. Despite the apparent promise of scenario-based methods for situated evaluation, relatively little research has demonstrated the efficacy of these techniques in CSCW. Haynes et al **[Haynes et al, 2004]** report on the use of scenario-based methods for evaluating collaborative systems. Scenario-Based Evaluation (SBE) provides evaluators with realistic settings in which to base their evaluations. Evaluators perform semi-structured interviews of the users to discover scenarios and claims about them. Then, focus groups validate these findings. The frequency and percentage of positive claims help quantify the organizational contributions of the system, and the positive and negative claims about existing and envisioned features provide information to aid in redesign.

In order to enhance the efficiency of usual scenario the Cooperation Scenarios Evaluation (CSE) **[Stiemerling and Cremers, 1998]** method aims to capture as much of the relevant context as possible in cooperation scenarios, especially the motivation, the goals, but also the workload of the different participating roles. Cooperation scenarios are context-rich, informal, textual descriptions of cooperative activities. They also contain the goals and subjective opinions (e.g. trust) of persons and other possibly relevant contextual elements. In order to construct scenarios, evaluators conduct field studies, semi-structured interviews, and workplace visits. They use scenarios for evaluation in three stages of the design process (see Figure 7):

- *Evaluation of scenario validity*: First they exchange cooperation scenarios among designers after write-up and critically compare and discuss them, specifically asking questions like "I did not get this detail. Who said that and are you sure, you've interpreted it correctly?". This first step is done after the design of any new functionality.
- *Theoretical evaluation of system design*: After having an early stage of the new design, they insert the new design, i.e. the new CSCW functionality, in the scenario and analyse for each role in the cooperative activity how individual part of the task change.
- *Practical evaluation in user workshop*: Then they conduct a feedback workshop, during which they presented an early version of the prototype to end users to discover design flaws.

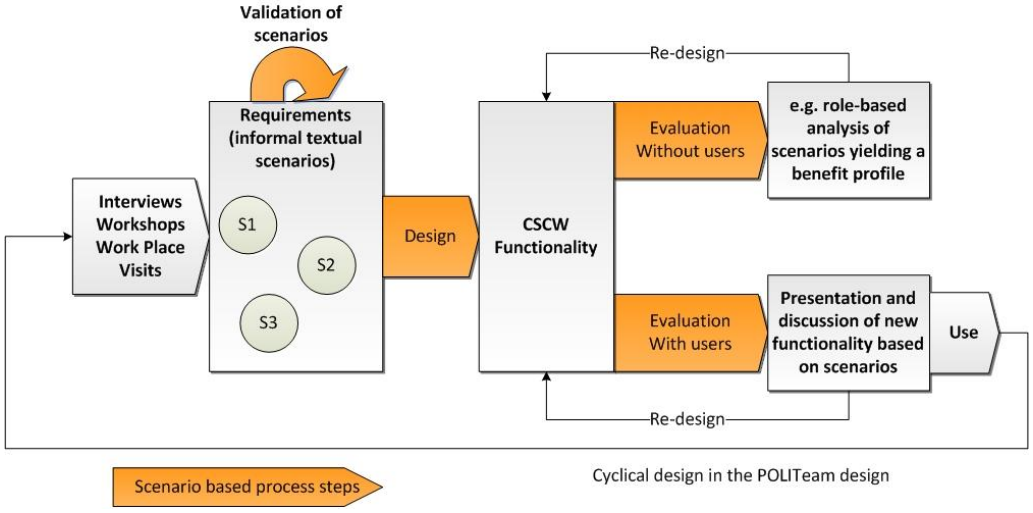

Figure 7 Cooperation scenario method [Stiemerling and Cremers, 1998]

The main advantage of this method lies in the fact that at least some contextual factors can be taken into account early in the design process when creating innovative

functionality and envisioning its effects on CSCW. Indeed, as you partially base and refine your design on the analysis of scenarios to be supported by the system, you can regulate your development and better adapt your design to the actual needs and a more precise consideration of the context. On the negative side we can mention the misunderstanding and misinterpretations of issues raised and in a second time the divergence in the different designers' ideas about the users and the fields of application.

As an example of real collaborative system evaluation through scenario-based method, Lau et al **[Lau et al, 2003]** have presented a case study on the evaluation of KiMERA (Knowledge Management for Enterprise and Reach-out Activities). The methodology consisted of four phases:

- *Preparation*: designing the scenarios/ episodes which would be close to the usage in real life; defining the role and providing enough guidance for the participants to role-play; ensuring the essential aspects for evaluation will be captured in a format that can be analysed later; and paying attention to constraints of time and resources.
- *Data Collection*: only involved for making logistic arrangements and hands-off monitoring.
- *Analysis*: during this phase, the issues raised were summarised from the data collected and the support team documented their responses and suggested actions against each issue.
- *Feedback*: provided the fruit of the effort in the evaluation exercise, suggested enhancements were split into training and/or development issues so they would be routed to the right places for action. Unresolved issues were also flagged so they were not forgotten.

Finally Antunes et al [Antunes et al, 2005] propose an advanced analytic method based on validated engineering models of human cognition and performance (GOMS **[Card et al, 1983]** and its family of models, such as GOMSL **[Kieras, 1996]**) to evaluate the usability of groupware systems: Analytic Evaluation of Groupware Design (AEGD).

GOMS (Goals, Operations, Methods and Selection Rules) is based on a cognitive architecture (user and physical interface) and a set of building blocks (goals, operators, methods and selections rules) describing human computer interaction at a low level of detail.

AEGD method is divided in three steps:

- *Step 1*: Defining the physical interface: the first step consists in characterizing the physical interface of the groupware tool under analysis. Considering the complexity of many groupware tools, the physical interface may be divided into several shared workspaces. The outcome of this step is then: a list of shared workspaces; definition of supported explicit communication, feed through, and back-channel feedback information; and characterisation of supported coupling mechanisms.
- *Step 2*: Breakdown definition of critical scenarios: the second step describes the functionality associated to the identified shared space with the respect to critical scenarios. Alternative design scenarios may be defined, considering several combinations of users' actions.
- *Step 3*: Comparing group performance in critical scenarios: the final step is dedicated to compare the alternative design scenarios defined in the previous step. The predicted execution time is selected for these comparisons. The authors utilize KLM (Keystroke-Level Model) **[Card et al, 1980]** to predict execution times.

This analytic evaluation method can be used to quantitatively predict and compare the usability of shared workspaces, without requiring users or the development of functional prototypes. Nevertheless it has two limitations **[Antunes et al, 2006]**: first a narrow-band view about collaboration restricted to shared workspaces and their mediation roles. Second, the method is limited by the selection of critical scenarios.

The results of the SBE method suggest that scenario-based evaluation is effective in helping to focus evaluation efforts and in identifying the range of technical, human, organizational, and other contextual factors that impact system success. The method also helps identifying specific actions such as prescriptions for design to enhance system effectiveness. Furthermore, basing an evaluation on scenario can be really helpful for structuring the evaluation itself; it can draw basic phases of evaluation. However, this kind of method is somewhat less useful for identifying the measurable benefits gained from a CSCW implementation due to the potential complexity to determine what part of the system has improved the collaboration. Despite, we think that if you accurately write your scenario keeping in mind the focus of your evaluation you surely can evaluate precise features with few disturbances.

The usefulness of scenarios in evaluations would depend on the design of those scenarios and how they would be used with other techniques.

3.1.2. Original CSCW Evaluation methods

Traditional methods had some advantages, firstly they existed, secondly people had already used them, and so they can have better feedbacks. Despite these advantages, those methods do not handle all aspects of CSCW and are not suited to evaluate them completely. That's why researchers started to develop new methods, natively designed for CSCW evaluation. There are a handful of evaluation methods that were designed specifically for the CSCW domain.

In the following we propose to study original CSCW evaluation methods according to several relevant aspects of this domain: functionalities required for the support of the collaboration (meeting support, file sharing, messaging, etc.), features to simplify the collaboration (such as automatic meeting summary), efficiency of the collaboration (has the collaboration been carried out in the expected time, with the expected results ?), usability of the system (is the system user-friendly or does it require intensive actions for users, disturbing them and making them loose precious time ?), knowledge management (is knowledge properly handled by the system, do collaborators have appropriate access to knowledge, and is it channel efficiently ?). Even if all methods don't suit the proposed aspects, we think they nicely reflect the methods analysed.

3.1.2.1. Discount Methods

In a first time we study methods which aim at providing a low-cost evaluation methodology. Indeed, one of the sources of reluctance of developers to use complex evaluation method is their high cost. Given that, it is completely natural that several researches focus on the development of low cost methods for evaluation. As we have seen in the previous section even adaptation of classical discount methods were made to provide a low cost evaluation of CSCW systems. We'll now present some of the discount methods developed especially for the evaluation of collaborative systems.

MITRE's Evaluation Working Group Methodology **[Drury et al, 1999**] or ECW was developed to give the group a timely, low-cost technique for evaluating CSCW systems for DARPA. The methodology has two phases.

Phase 1: in this phase, the CSCW system is classified according to a CSCW framework developed by ECW (Figure 8). The framework consists of four broad categories: requirements, capabilities, service, and technology. The requirements level specifies tasks the users will perform. Tasks include work tasks like editing a document, transition tasks: passing a document onto a reviewer for comments, and social protocols like coming to a consensus about who controls the floor during a discussion. The capability level describes the functional components a CSCW system has to support tasks inherited from the requirements level. For example, two systems might support shared editing; however, one allows synchronous editing while the other is strictly asynchronous. The service level describes the general types of

applications available to support collaborative activity. Examples include e-mail, audio, video, and remote windowing. The technology level describes a specific implementation of a service.

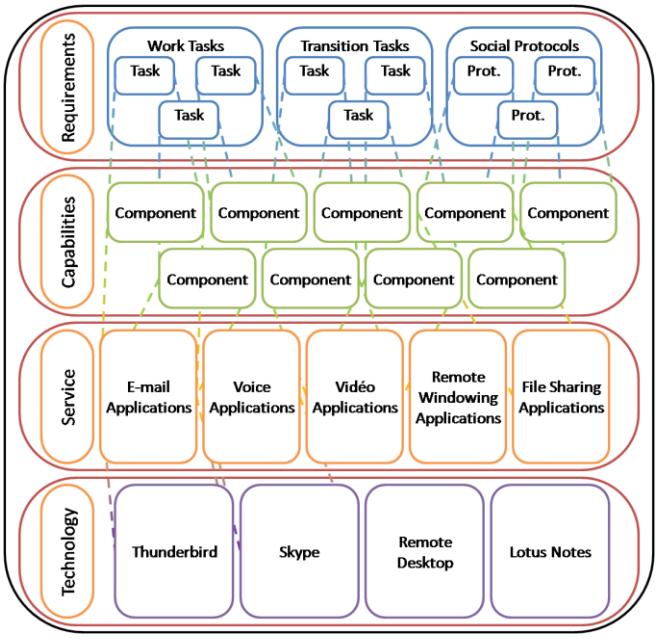

Figure 8 ECW First Phase Framework

Phase 2: uses pre-written scenarios to evaluate a CSCW application. Scenarios are not application specific rather they are derived from the categories and sub-categories in ECW's framework. The classification of the application in phase one identifies the scenarios to use in the second phase. Scenarios are supported at each level of the framework and come in two forms: unscripted and scripted. An unscripted scenario provides a general description of what a user is supposed to be doing. This freedom of action allows observers to determine how easy it is for users to complete a set of tasks. Scripted scenarios dictate exactly how a user will use the system to complete a set of tasks. The rigid structure allows observers to compare similar applications, measure the effectiveness of the same application with different groups, and reproduce user activity.

ECW is noticeable as it furnishes basic instruments to easily compare two applications of the same domain (for instance two web browsers of two instant messaging applications). This can be particularly relevant if you develop a system with already existing competitors. Besides, we think it can help designing your evaluation processes even if you're developing a new system. By considering systems close to your own, you can infer your own scenarios.

ECW has an obvious characteristic; it does not rely on users' feelings or opinion. On the exact opposite, Perceived Value **[Antunes and Costa, 2003**] is a low cost approach to evaluate "meetingware" centred on the measurement of users' opinions about the organizational impact of meetingware technology and the alignment between system capabilities and developers' and users' expectations. Measuring perceived value requires a negotiation between developers and users. It is composed of the following steps:

- *Step 1*: identifying the components that are relevant to evaluate the meetingware system. The developers execute this task before the first contact with the users, based on their appreciation of the deployed functionality.
- *Step 2*: identifying the concrete external attributes that will be evaluated by users. The selected list of attributes should be negotiated between the developers and users.
- *Step 3*: having users experimenting the system. This can be done in several ways. The authors choose to do it with "hands on" meetings.

- *Step 4*: requesting users to analyse the contribution of the meetingware components to the selected list of attributes and fill out the evaluation map. This task may be accomplished either individually or as a group, in a meeting discussion.

This method is interesting, but one point restricts its flexibility, by defining the evaluated features before the evaluation users are not able to evaluate an unexpected aspect of the system. Nevertheless, this method can be really efficient to evaluate early developments of a precise feature.

Evaluation Working Group (EWG) methodology **[Damianos et al, 1999]** offers a different approach to CSCW evaluation by helping people finding appropriate methods for their evaluation. Besides, it was developed with the goal of providing a reliable but inexpensive means of evaluating collaborative software tools.

EWG **[Damianos et al, 1999]** methodology approach consists in the following steps:

- *Step 1*: *Formulate hypotheses*: hypotheses are the driving factor for determining all other aspects of conducting an evaluation;
- *Step 2*: *Determine appropriate evaluation method(s) and scope*: determine evaluation goal, then choose an evaluation method (different types of evaluation and the associated questions addressed are shown in Tab 4) and determine the resources required to carry it out.
- *Step 3: Select data collection instruments and identify measures*: Data collection instruments are the means of taking both direct measurements (measuring individual actions) and indirect measurements (making inferences from the responses to a questionnaire or from measurements taken). Data from the evaluation can be collected by: logging tools, direct observation, questionnaires/interviews, video and audio recordings.

This methodology has an interest from the evaluation preparation point of view. Indeed, instead of providing a complete method for evaluation it gives advises on which kind of method to use for the evaluation.

Question	Type of evaluation
Does the system run, and can we afford it?	Feasibility evaluation
Have improvements been made to a system?	Iterative evaluation
Is system A better than system B?	Comparative evaluation
Does the system have the necessary capabilities?	Appropriateness evaluation

Tab 4 Evaluation Questions [Damianos et al, 1999]

Compared with traditional discount methods applied to CSCW, original discount ones share some advantages: cost limitation, relatively fast process. Still, original ones suffer from fewer drawbacks as they are more closely coupled with the CSCW domain.

3.1.2.2. Multiple-phased evaluation

In a second time we have to point out an emerging aspect of collaborative system evaluation, the fact that more and more of them are designed in multiple phases having for objectives to facilitate and improve evaluation. Among them we can cite MITRE **[Drury et al, 1999]** which has been evoked previously.

The Participatory Evaluation Through Redesign and Analysis Methodology **[Ross et al, 1995]** or PETRA was Ramage's first attempt at an inclusive CSCW evaluation methodology. For PETRA the evaluation should incorporate multiple perspectives: evaluator and users. The evaluator will be interested in theoretical models of collaborative activity induced by the system. Users, on the other hand, will be preoccupied with the design of the system and how they feel as a participant.

The System Evaluation Stakeholder Learning Methodology **[Ramage, 1999]** or SESL is an improvement of the pioneer PETRA. Like PETRA, SESL recognized the need for

multiple evaluation methodologies. The "perspective" concept in the evaluation was replaced by the "stakeholder" concept. Ramage recognized that there were many different types of stakeholders involved in the evaluation of a CSCW system. The view of a stakeholder influences the evaluation. A software developer, for example, may consider the system good if it doesn't have any detectable bugs. A psychologist may consider the same system bad because the floor control policies create tension between the participants. A manager using the system might dislike it because the floor control policy makes it difficult to control employees. Employees might like the system because it allows them more freedom of expression. This idea simply refers to the "domain" of each actor in the system. The simple fact that the system fulfil requirement doesn't implies that requirements match business expectations. In this case, it falls to business architects and engineers to adapt the requirements for software architect and developers to take care of them.

The idea of learning was also introduced in the evaluation. Ramage believed that the evaluator, as an active participant, could both teach and learn from the other evaluation participants. With this last idea, we can point out the fact that for the SESL methodology, the evaluation becomes a collaborative activity where each participant brings his point of view with his experience and skills.

Turning to more generalist methods, one of them particularly attracted our attention; Knowledge Management Approach (KMA) **[Vizcaino et al, 2005**] is a method for evaluating collaborative systems from the knowledge point of view, taking into account whether the tools helps users to detect knowledge flows, to disseminate them, to store previous experience and to reuse it. The knowledge circulation process is comprised of six phases:

- *Phase 1*: Knowledge creation: it occurs when new information is obtained and understood;
- *Phase 2*: Knowledge Accumulation: new collaborative systems try to learn from current client use and automatically store information that could be needed in the future;
- *Phase 3*: Knowledge Sharing: this sharing often creates communities called "communities of practice" where each member cooperates by sharing knowledge about a common domain;
- *Phase 4*: Knowledge Utilization: depends on two constructs: degree of knowledge utilization in an organization, and knowledge utilization culture;
- *Phase 5*: Knowledge Internalization: is related to three constructs: capability to internalise task-related knowledge, education opportunity and level of organization learning

To perform evaluation, each phase has a list of associated questions, which may be used as a checklist by evaluators.

Knowledge management is an "all mighty" paradigm in information technologies, it has the ambition to ascend collected information to a higher form of existence, making them meaningful and reusable to provide a base on which you can rely. This can be done by the long process of experiment and feedbacks. This approach is doubtlessly an efficient one, but it only provides a domain-independent method for evaluation and it has to be coupled with some more concrete ones or at least to be adapted for a specific project.

If you're looking for something more concrete, CAMELOT (CSCW Application MEthodoLOgy for Testing) **[Dugan et al, 2002]** can surely help you, it merely is a technology-focused methodology for testing CSCW software. CAMELOT provides an organized set of specific techniques that can be used for technological evaluation. CAMELOT decomposes CSCW application into four intersecting software technology as shown in Figure 9:

- *General Computing*: describes software components that provide general application capabilities.
- *Human Computer Interaction*: describes components that deal with interface between the user and the software system.
- *Distributed Computing*: describes components that are responsible for multitasking and multiprocessing in the application at the thread, process, processor, and machine levels.
- *Human-Human Interaction*: describes components that facilitate interaction between users during application use.

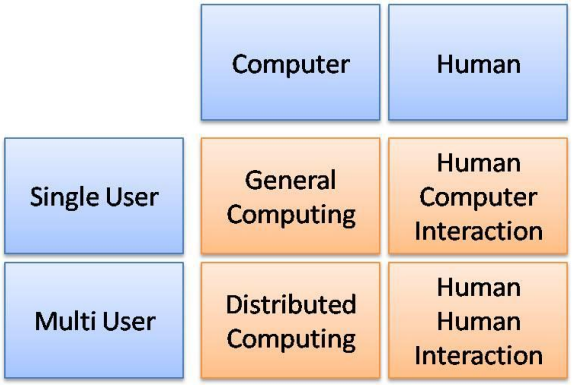

Figure 9 Intersecting CSCW technologies [Dugan et al, 2002]

CAMELOT is applied in two stages:

- *Stage 1*: single user Evaluation is subdivided into general computing and human-computer interaction testing. General Computing encompasses testing techniques that can be used with any kind of software application, Human Computer Interaction techniques concentrate on identifying problems with application's user interface.
- *Stage 2*: multi-user Evaluation is decomposed into distributed computing and human-human interaction. Testing Distributed Computing focuses on the multi-thread, task, processor, and machine challenges that occur in CSCW application. Human-Human Interaction concentrates on testing the software components that facilitate interaction between users.

In contrast with other methods, this approach has a technological focus; it should be used in conjunction with other methodologies for a complete evaluation of a CSCW system. Despite its lacks this method is particularly noticeable as it points out a fact that we haven't considered yet: the need to evaluate CSCW systems for the multiple users' point of view, but also from a single user one. Indeed, even if your target is the collaborative work, it must not be done at the cost of the single user experience. That is to say, designing an efficient collaborative system requires that all single user aspects have to be at least as ergonomic as collaborative ones.

Similar in form but with a different consideration of collaborative software, CSCW Lab **[Mendes de Araujo et al, 2002]** defines a method for groupware evaluation comprising the steps for conducting the evaluation and guidelines for using any technique and instruments. They identify four inter-dependent dimensions (or concepts) for groupware evaluation as shown in Figure 10. The authors considered each dimension as the subsequent steps of the method. According to the authors, groupware evaluation can address the context in which a certain application is (or will be) used, its usability, the level of collaboration achieved through its use and its cultural impact.

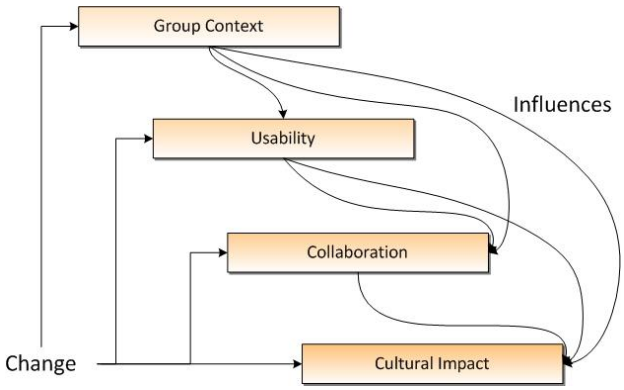

Figure 10 Dimensions for groupware evaluation [Mendes de Araujo and al, 2002]

Since every single group constitutes a unique entity, it is important to provide a sharp characterization of the group context, in order to study different effects of a certain technology on different kinds of groups more precisely. Furthermore, group characteristics can influence other evaluation dimensions: high commitment can make a group overcome usability problem, and increase its level of collaboration and the positive impact of a tool on the work environment. Usability influences an application's use and acceptance. In the case of groupware, this means it affects the level of collaboration achieved through the use of a tool and its impact. Apart from group context and usability, a group's collaboration level is influenced by several factors, among them the nature and objectives of the group. The level of collaboration reached within a group can then affect the cultural impact of the used tool. This approach is really pleasant, by its layered design and by its fine analysis of tools impact.

As we have heavily hammered, evaluating a CSCW system is a complex task involving the analysis of multiple users' point of view evolving in a distributed computing environment. As any complex system, it has to be considered from multiple aspects to be efficiently understood and analysed. Thus, more and more of recent evaluation methods are designed to propose several steps in the evaluation of systems. However, each method has its own approach of evaluation and they do not all propose the same steps. While KMA proposes six steps representing the evolution of Knowledge "life", CAMELOT splits the evaluation in two distinct phases: single and multiple users. There is no doubt that future evaluation methods will be composed of multiple phases and a way to improve it could be to mix different existing approaches.

3.1.2.3. Task Model Based Evaluation

In this third section we consider a well-known concept in collaborative work: task-models. These models provide a way to represent tasks performed by the different users of the system. They can be used for different purposes: evaluating the role of a user inside the collaboration, triggering some events at some point of the task, making statistics about the tasks, feeding a context manager or else. This concept is really popular among the CSCW community and we found it particularly relevant and useful to look for evaluation methods relying on it.

Once again we have to refer to MITRE **[Drury et al, 1999]** which uses tasks as the basis to define and describe components, services and technologies to evaluate.

With a more confined consideration of collaboration work, Collaboration Usability Analysis (CUA) **[Pinelle and Gutwin, 2003]** is a task analysis technique designed to represent collaboration in shared tasks. It is focused on the teamwork aspects of a collaborative situation. It provides high-level and low-level representations of the collaborative situation and task to be studied, and multiple ways to represent actors and their interactions. CUA is based on a hierarchical task model that represents the procedural elements of a group task in a shared workspace. The task hierarchy includes scenarios: to describe the high level context of the collaborative situation, tasks: to indicate specific goals

within the scenario, and low-level task instantiations: to represent individual and collaborative ways of carrying out the task and actions: common ways to carry out the mechanic specified in the collaborative task instantiation.

CUA proposes that each collaborative action can be mapped to a set of mechanisms of collaboration, or fine-grained representations of basic collaborative actions, which may be related to elements in the user interface. The resulting diagrams capture details about task components, a notion of the flow through them and the task distribution. CUA is maybe one of the most promising approaches for CSCW evaluation that we have been aware of. The fine description and evaluation of task and subtasks is, for us, a key element to a proper evaluation. However, this method absolutely requires the development of a framework to help people lead their evaluation with this highly-descriptive method.

Still, many other works based on task models have been conducted, for instance Baeza-Yates and Pino **[Baeza-Yates and Pino, 1997**], **[Baeza-Yates and Pino, 2006**] made a proposal for the formal evaluation of collaborative work: Performance Analysis (PA). The application to be studied is modelled as a task to be performed by a number of people in a number of stages. Stages can be time-based (days), location-based (meetings) or task-based (sub-projects). The goal of authors is to measure the efficiency of the task performed by the group. They define the following concepts.

- *Quality*: how good is the result of collaborative work?
- *Time*: total elapsed time while working.
- *Work*: total amount of work done.

The evaluators must define a way to compute the quality (e.g., group recall in a collaborative retrieval task), and maximize the quality vs. work done either analytically or experimentally.

The applicability of this type of formal analysis is limited by the availability of quantitative data concerning the application, which in the case of collaborative software can be complex to collect and even more to interpret correctly.

Task model is a powerful concept, lightly used it can provide information on the progress of a task, task participants, estimations on the time to complete and else. Heavily used, it has the capacity to provide almost any information concerning the collaboration. In the specific use of evaluation it can provide accurate information about elapsed time to perform an action, actions realised to complete a specific task, estimate communication flows among participants and else. Thus, wisely used, tasks model are able to furnish to evaluators almost all information they can wish about the collaboration and the effectiveness of the system. Still we think that relying only on the task model is insufficient, it surely can represent how well the system works, but it can't tell how users feel about the system and the supported collaboration. That's why we think a task-based model has to be paired with a consideration of users' experience.

3.1.2.4. "Unconventional" methods

Even if most of the methods are more or less closely related, some of them are particularly hard to put in a specific category and don't follow usual aspects. Thus, due to their approach or point of view some of these have to be considered individually.

First in this category: "Quick and Dirty Ethnography" (QDE) **[Hughes et al, 1994]**. Ethnography refers to the qualitative description of human social phenomena to produce detailed descriptions of the work activities of actors within specific contexts. The phrase "Quick and Dirty" does not refer simply to a short period of fieldwork but also signals its duration relative to the size of the task **[Hughes et al, 1994]**.

QDE aims to adapt ethnography to evaluation. Evaluators do brief workplace studies to provide a general sense of the setting for designers. QDE accepts the impossibility of

gathering a complete understanding of the setting, providing a broad understanding instead. It suggests the deficiencies of a system, supplying designers with the key issues that bear on acceptability and usability, thus allowing existing and future systems to be improved.

This method is often used as a precursor to other ethnographic research methods. It can be useful in increasing awareness of large-scale usability and acceptability issues that are important in the design of a new system. However, it is frequently the only form of ethnography practiced due to imposed time and/or budget constraints. It can yield valuable knowledge of the social organization within a work setting in a short amount of time relative to the size of the project. Despite, the results are limited to a general understanding of a work culture. It is possible that a quick and dirty ethnographic study can lead to a false sense of understanding for a working culture.

Ethnographic methods are effective for explicating contextual factors, but require longitudinal, immersive data collection efforts to produce useful results, though some suggest that the time and expense of ethnographic methods can be reduced if researchers go into the field with specific research questions pre-defined **[Hughes et al, 1994**]. Besides, ethnographic approaches can be utilized to evaluate groupware, but these techniques require fully functional prototypes, which are expensive to develop.

Last of all, E-MAgine (EMA) **[Huis in't Veld et al, 2003]** is a method based on two concepts: Contingency Perspective (systems should match their environment in order to thrive and to be effective) and ISO-norm for ICT tools (Quality in Use). The method is divided in two main phases (Tab 5). The first phase provides a profile of the group and the groupware application. The first step of the evaluation begins with a meeting between client and evaluator, in which the goals are set. Then, a quick semi-structured interview with someone familiar with the group is applied to build a profile of the group and scenario (step 2). In the third step more aspects may be selected for further evaluation, such as social cohesion, usability issues, and trust among group members. A selection of evaluation instruments is done (step 4). Step 5 provides a final profile of the fit between the group and its applications based upon the test results of the applied evaluations instruments. Finally, in the step 6, the results are fed back to apply the proposed changes.

		What	Whom	Result
Phase 1	Step 1	Inventory	Client and Evaluator	Evaluation plan
	Step 2	Apply Quickscan	Evaluator and Group leader	Filled out Quickscan
Phase 2	Step 3	Feedback of first level profile	Evaluator and Group leader	Choice of evaluation instruments
	Step 4	Apply instruments	Evaluator and Group members	Evaluation data
	Step 5	Construct Final Profile	Evaluator	Final Profile
	Step 6	Feedback workshop	Evaluator and Group members	Finished evaluation

Tab 5 Steps of E-MAGINE

This method can surely be efficient. However it is not completely adapted to evaluate CSCW systems. Indeed, it hardly focuses on the group concept, which in some case can be complex to identify or even does not really exist.

3.2 Existing taxonomies of evaluation methods

As seen in the previous section, many evaluation methods have been proposed and designed to evaluate CSCW systems. Despite this profusion of methods it's not always easy to know which method you should use for your evaluation. Solving this issue requires to find a common ground to organize these methods, in this perspective we will now present the already existing taxonomies proposed in the literature.

As a critical point and a bottleneck in term of CSCW evolutions, the evaluation has been an interesting but also one of the most complex research domain. In order to make it more understandable to mankind, several researchers decided to go above the evaluation domain and tried to organize the work that have been done by others. Among them, we can cite Randall **[Randall et al, 1996**]. They have identified four orthogonal dimensions to classify the kinds of evaluation in groupware:

- Summative vs Formative;
- Quantitative vs Qualitative;
- Controlled Experiments vs Ethnographic Observations;
- Formal and rigorous vs Informal and opportunistic.

The authors state that the most used types of evaluations are the summative controlled and experimental (considered as a formal technique); and the formative qualitative-opportunistic approaches (considered as an informal technique). Usually a distinction is made between formative and summative evaluation. Formative evaluation is meant to inform designers and developers designing the service or application and getting user feedback about preliminary versions. Summative evaluation is meant to inform the client or the external world about the performance of the service or application in comparison to a situation where there is no such service available, or to a previous version, or to competing services; in brief, to demonstrate the usefulness of the system.

This first taxonomy is relevant as it allows considering any evaluation method. However, for a proper classification, we should be able to consider methods as intervals over the specified dimension. Indeed, as we can vary methods settings, their classification can't be limited to a precise point in Randall's space.

CSCW evaluation is a vast domain, and as such it can be fathomed from many perspectives. In **[Ramage, 1999]** the author proposes to consider it from five aspects as shown on Tab 6.

Ethnography	Qualitative	Psychological	Systems Buildings	Taking Advice
• Ethno-methodology • Conversational Analysis • Interaction Analysis • Distributed Cognition • Activity Theory • Breakdown Analysis • Others	• Interviews • Questionnaires • Group Discussion	• Lab Experiments • Analytic Approaches • GOMS Approach	• Iterative Prototyping • Participatory design • Beta Testing • Heuristic Evaluation • User Testing • Semi-Situated Ethnography	• Consumer Reports • Consultancy Reports • Marketing literature

Tab 6 Ramage evaluation methods taxonomy

- *Ethnography* is the study of an entire organization in its natural surroundings over a prolonged period of time.
- *Qualitative methodologies* ask people questions about their experiences and compare/ contrast the answers with other people surveyed.
- *Psychological methods* use either lab experiments that focus on the isolation and analysis of a very specific phenomenon or analytic approaches that attempt to describe human interaction using formal models.

- *Systems Building focuses* on the development of partial or complete systems with the goal of improving them based on the evaluation.
- *Taking Advice* uses oral, video, and written information about an application as an evaluation mechanism.

Ramage points out that the very nature of CSCW imposes his taxonomy to be imperfect. As this nature results of the intricate combination of various disciplines, it makes it even more complex to provide a unified taxonomy of evaluation methods. Also, given the breadth of his taxonomy there is some overlap between the methodologies.

David Pinelle and Carl Gutwin **[Pinelle and Gutwin, 2000]** have an approach closer from Randall's as they consider CSCW systems from strictly unrelated aspects. They reviewed forty-five CSCW articles from 1990 to 1998 with the objective to evaluate the use of evaluations methods and their different categories. They classified the evaluations both in relation to the environment where they are accomplished (natural occurrence or simulation of the phenomenon), and the degree of the variables manipulation (rigorous or minimum control of variables). See Tab 7.

		Manipulation	
		Rigorous	***Minimal/None***
Setting	***Naturalistic***	Field Experiment	Field StudyCase Study
	Controlled	Laboratory Experiment	Exploratory

Tab 7 Evaluation classification [Pinelle and Gutwin, 2000]

They report that most of the articles only include laboratory experiment or even no evaluation, only some articles provide an experiment in real settings. We can also point out the fact that the evaluation is not completely integrated into the development process, despite almost every book, article, teacher or expert laud it. Indeed, most of evaluation processes are held for a finished application, package or prototype, it is not yet a continuous task during the development process. Another part of this report states that the most classical mean for evaluation is the observation, with direct sight or videotapes. The second technique is composed of interviews and questionnaires. They point out that evaluations lack of interest for "organizational work impact" and that most of them only focus on "Patterns of system use", "Support for specified task", "User Interaction through the system", "Specific Interface features" and "User Satisfaction". They also point out the fact that the evaluation should include gradually more and more work settings during the development of the software. They stress the fact that it is really important to lead evaluation even at the beginning of the development, it can avoid serious problems or misunderstandings, allowing you to be sure of what you're doing and protect from "chain reactions", meaning that if you have created a part of the application without evaluation and when you finally test it in real conditions, users can tell you that "he didn't want to have a shared file storage, but a personal one", and then you can redevelop most of your application. Furthermore authors suggest that evaluations should be shifted around users and their organizations and those researchers should try to reduce time and cost of evaluations methods, making them more attractive for companies and researchers themselves. Their conclusion is that each work used different approaches, methodologies or techniques for conducting evaluations.

Recently, **[Herskovic et al, 2007]** have led an interesting survey on some evaluation methods. What they suggest is not to use only one evaluation method, but to divide it into three phases:

1. *Formative lab-based methods* (perform some pre-evaluation to avoid main errors).
2. *Field methods* (with the participation of users associated context).
3. *Qualitative methods in real work settings* (evaluate if it really works).

In order to facilitate the planning of evaluation Herskovic proposes a classification of evaluation methods depending on some simple, but still fundamental, characteristics with limited values:

- *People Participation* : can be users, developers, experts or any combination of them;
- *Time to Apply the Method*: the moment when the evaluation takes place (before, during or after the development of the application);
- *Evaluation Type* : describe if the evaluation is qualitative or quantitative;
- *Evaluation Place* : can be a laboratory or usual work place;
- *Time Span* : the time dedicated to the evaluation, it can be hours, days or weeks;
- *Evaluation Goal*: describes the purpose of the evaluation method, what it is aimed to. It can be the evaluation of the product functionality, the collaboration process of the system or the product functionality considering the collaboration context.

In addition to this first classification, Herskovic furnishes a second one to estimate the final cost of an evaluation method according to its characteristics. It is another tool facilitating the construction of the triple-phased evaluation process. This work is particularly relevant as it is based on the analysis of existing methods. This characterization is a good step in the long walk to a better understanding and appreciation of evaluation.

Finding a structure among evaluation methods is a necessary and complex task. From what we can have read until now, the most advanced and promising work is doubtlessly the one of Herkovic: the organization of evaluation in multiple phases and the clear description of relevant aspects of the evaluation are key concepts to know how to evaluate a CSCW system.

3.3 Existing CSCW evaluation Frameworks

If you want to evaluate your work you need a method, you need to prepare your evaluation but you can also need a tool to facilitate its process and allow you to make the evaluation in a shorter time with a higher efficiency. Evaluation frameworks in the literature fall into three different camps **[Neale et al, 2004]**:

- *Methodology-oriented;*
- *Conceptual;*
- *Concept-oriented*.

Methodology-oriented frameworks provide a useful support for CSCW researchers to know the possible types of evaluation. Still, they can't really help you find what method you should use. Furthermore as we have already presented evaluation methods, we won't dwell on this camp to go directly to conceptual and concept-oriented ones.

3.3.1. Conceptual CSCW frameworks

Conceptual CSCW frameworks describe the group factors that should be considered during evaluation for discerning what should be evaluated in CSCW. A number of conceptual frameworks have been proposed that outline the major factors relevant to analyse CSCW. The Olsons' proposition **[Olson et al, 2001]** is targeted toward analysis of the effects of video on distance interactions; they develop a nice framework that is generally applicable. The framework of Pinsonneault and Kraemer **[Pinsonneault and Kraemer, 1989**] has some similar framework development, but is targeted toward GDSS (group decision support systems.) These frameworks for small group interaction divide the context variable into classes having to deal with

- Characteristics of the group (group members and their relationships),
- Characteristics of the situation;
- Characteristics of the technology;

- Characteristics of the task.

They have several properties in common: Group, characteristics, situation factors (context), individual characteristics, task properties, group process, task and group outcomes. Each one of these factors can have a number of issues associated with them. Most of these frameworks stem from early research on group behaviour. The factors in these frameworks correspond generally to situation, task, and human considerations in any type of applied research endeavour.

On a similar approach but with a different perspective Neale et al **[Neale et al, 2004**] propose an awareness evaluation model that aims at providing a map of the variables and relations one should consider when evaluating computer supported cooperative work applications. Figure 11 shows the major variables considered: Contextual factors, Work coupling and communication, Coordination, Awareness and Common Ground.

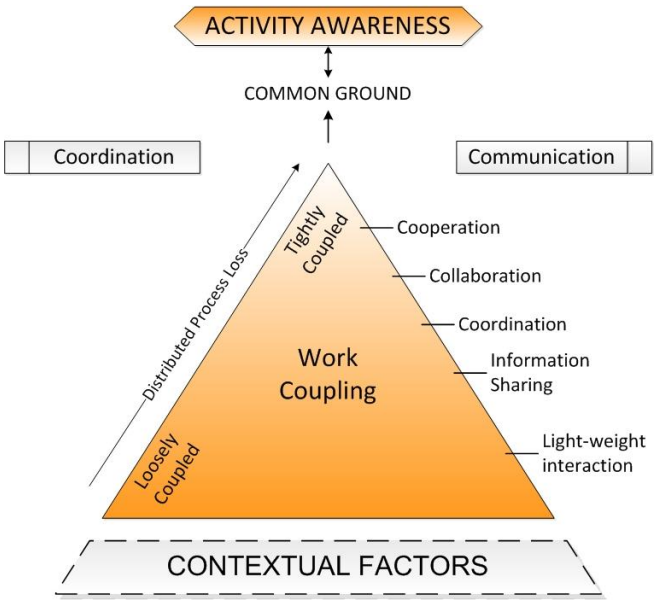

Figure 11 Activity Awareness Evaluation Model [Neale et al, 2004]

According to the authors, a higher work coupling (intended as the demand of the work for information sharing) entails a higher demand for coordinated behaviour and communication. Only when a system provides the necessary support for these needs can group members achieve the level of activity awareness critical to effective group functioning. In other words, work coupling (always constrained by contextual factors) constitutes the basis for the requirements a system should satisfy in order for groups to succeed. Therefore, CSCW evaluations should not focus on individuals' actions within a certain activity but on activities themselves, spanning across people and locations in a determinate context.

Neale et al. divided work coupling in five progressive levels, according to the levels of coordination and communication they require and related to different levels of interaction: Lightweight interactions, information sharing, coordination, collaboration and cooperation. Lightweight interactions are only partially related to the work itself, while information sharing often conveys important background issues concerning it. Coordination-level activities require people to coordinate their action and communication, and such requirements increase for collaborations, which involve group members with highly interdependent tasks working towards a common goal. Finally, cooperation demands the highest level of coordination and communication, and involves people characterized by common plans, shared tasks and shared goals, which put team's priorities over individuals' goals. Related to coordination and communication is awareness: communication can only take place if people are to some extent aware of each other, and the more aware people are, the less there is a need to coordinate their activities. When awareness is shared among different people (joint

awareness) it produces the common ground necessary to communicate, collaborate and coordinate.

However, there are other frameworks with different approaches. We can cite Gutwin and Greenberg **[Gutwin and Greenberg, 2000]**, **[Pinelle et al, 2003]**; they introduce a conceptual framework for developing discount usability evaluation techniques that can be applied to shared workspaces groupware. The framework is based on support for the mechanics of collaboration: the low level actions and interactions that shared manner, these include communication, coordination, planning, monitoring, assistance and protection. The framework also includes gross measures of these mechanics: effectiveness, efficiency and satisfaction. The underlying idea of the framework is that some usability problems in groupware systems are not inherently tied to the social context in which the system is used, but rather are a result of poor support for the basic activities of collaborative work in shared spaces.

From a drastically different consideration based on a wish for a better integration of evaluation in the development process, Huang **[Huang, 2003]** presents an evaluation framework within lifecycle and stakeholder-oriented approach to support development of a virtual enterprise.

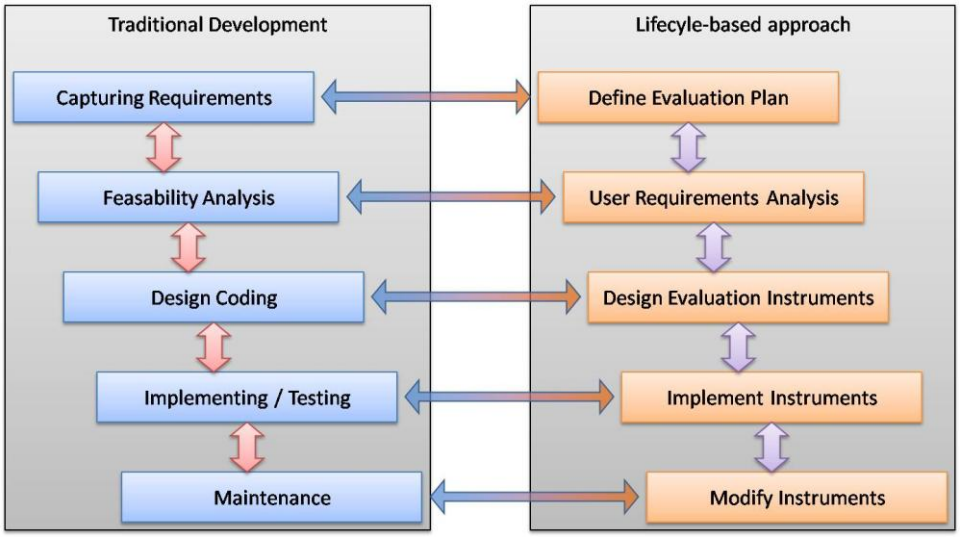

Figure 12 Life-cycle based approach [Huang, 2003]

According to authors a traditional developmental lifecycle is an iterative process evolving from feasibility and user requirements capture through initial specification and definitions, then to the feasibility analysis, the designing step, development/testing and last but not least the maintenance of the final product. From this lifecycle-based approach derives a sequence: defining an evaluation plan, user requirements analysis and specification, designing evaluation instruments, implement and modify the evaluation instruments. This approach is shown in Figure 12.

Providing an even greater degree of liberty of settings and configuration, Damianos et al **[Damianos et al, 1999]** have developed a framework for describing CSCW systems, metrics for evaluating the various components of a CSCW system, and a scenario-based evaluation technique. The goal of the framework is to facilitate description of a collaborative system and to evaluate how well that system supports various kinds of collaborative work.

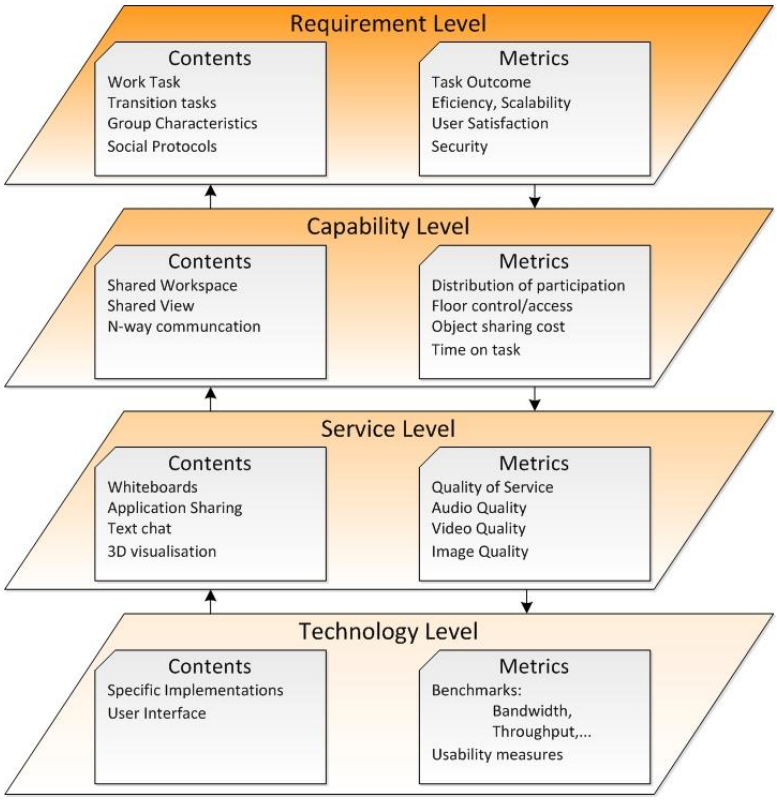

Figure 13 The four levels framework [Damianos et al, 1999]

As shown in Figure 13, the framework is divided into four levels:

- *Requirements*: this level describes the requirements of the group with respect to the tasks being performed and the support needed by the characteristics and social protocols of the group. It is divided into four sections: work tasks, transition tasks, social protocol requirements, and group characteristics;
- *Capability*: describes functionality that is needed to support the requirements. This includes ability to share documents and programs, support for awareness of other people and activities in the workspace, communication;
- *Service*: describes specific services that can be used to provide the capabilities needed for collaboration, e.g. e-mail, chat facility, telephone connections, and collaborative space management;
- *Technology*: describes specific implementations of services.

In phase with task-based models evoked earlier, Van Welie et al **[Van Welie et al, 1999]**, **[Van der Veer and van Welie, 2000]** have developed a usability framework focused on task specification relying on ethnographic observation. Figure 14 sketches this framework. On the highest level stands the ISO definition of usability, giving the three pillars for looking at usability which are based on a well-formed theory. The next level contains a number of usage indicators that can actually be observed in practice when users are at work. Each of these indicators contributes to the abstract aspects of the higher level. For instance, a low error-rate contributes to a better effectiveness and good performance speed indicates good efficiency and hence it can be an observable goal for design. The usage indicators are measured using a set of usability metrics.

One level lower is the level of means that can be used in "heuristics" for improving one or more of the usage indicators and are consequently not goals by themselves. For instance, consistency may have a positive effect on learnability and warnings may reduce errors. On the other hand, high adaptability may have a negative effect on memorability while having a positive effect on performance time.

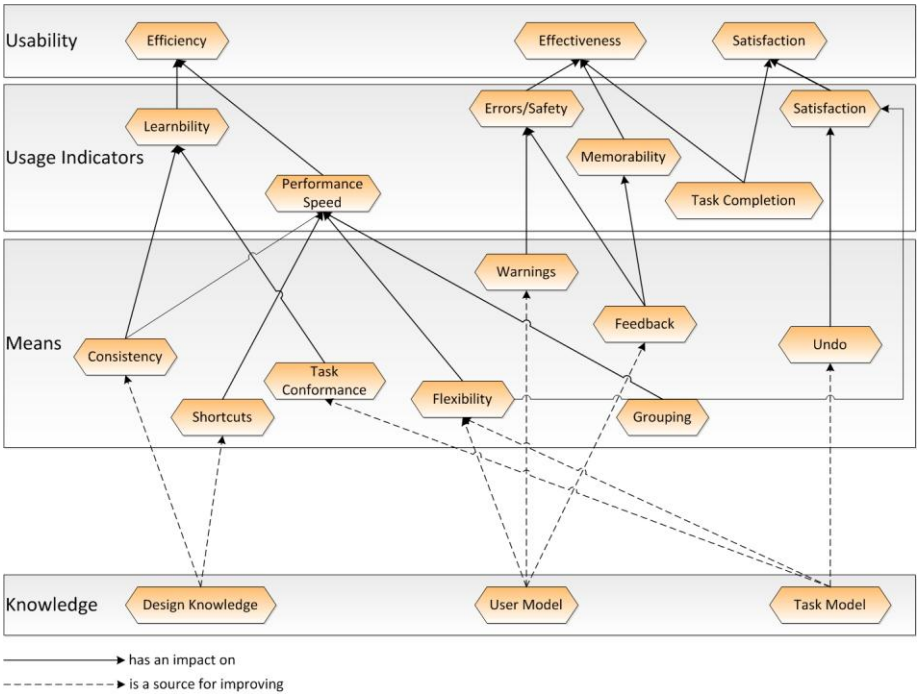

Figure 14 A Usability Framework [Van Welie et al, 1999]

Each means can have a positive or negative effect on some of the indicators. These means need to be "used with care" and a designer should take care not to apply them automatically. The best usability results from an optimal use of the means where each means is at a certain "level", somewhere between "none" and "completely/ everywhere/all the time". In order to find optimal levels for all means, the designer has to use the three knowledge domains (humans, design, and task). For example, design knowledge like guidelines, should include how changes in use of the means affect the usage indicators.

Conceptual frameworks are interesting as they provide a way of evaluating any collaborative system by analysing them with the same model. As we have seen this kind of framework is particularly effective if you want to evaluate a groupware without focusing on a precise feature, still it doesn't implies the objectivity of the evaluation as all conceptual frameworks consider CSCW systems from their specific model. By putting all systems on the same level, conceptual frameworks can also be used to compare and thus class and order collaborative systems. A natural inconvenient of this type of framework is its relative lack of accuracy and refinement when it comes to consider specificities of each system.

This inconvenient of conceptual frameworks is relative, and it does not mean they can perform an effective evaluation of system, but if your quest is to know how well your system handles a really specific aspect of collaboration or even a precise characteristic such as "photo sharing quality" you will need either a conceptual framework with specific evaluation criteria or rather an evaluation framework centred on collaborative media sharing evaluation or something close to it. This lead us to the second part of this section, concept-oriented frameworks, which on the opposite of conceptual ones offer the possibility to isolate the evaluation of characteristics with an higher precision and reliability.

3.3.2. Concept-oriented frameworks

Concept-oriented frameworks describe how specific methods can be used to measure concepts like communication effectiveness, awareness, or trust. They focus on specific aspects of group behaviours or concepts, such as communication or coordination. These frameworks are more limited, but they offer specific advice for focusing on limited or isolated aspects of group interaction.

Data logging has been a standard, however underutilized, software evaluation technique for single-user systems. Helms et al **[Helms et al, 2000]** extended these traditional logging approaches to collaborative multi-user systems, showing how data captured at a higher level of abstraction can categorize user-system interaction more meaningfully. They developed a three-tiered model to characterize the process and use of data logging. This model can be useful for understanding how logged data must be captured, transformed, and fully utilized. This model consists of three processes that iteratively raise the data to higher levels of abstraction, providing more meaningful information at each stage. First-order raw events, or capture-level processes, must capture user behaviour from system-level events, not all of which are of interest to the usability engineer. Second-order transformed actions refine the raw data by filtering and formatting the logged data for human readability or statistical analysis. This step includes removing irrelevant events, reformatting, and adding information, allowing data to be used in both quantitative and qualitative analysis. Third-order, or activity-level transformations, combine transformed data produced by second order processes with other usability data to create a more meaningful transcript of user sessions. Examples of other usability data include video recordings of group sessions, contextual inquiry, student and mentor interviews, screen capture, survey data, and think aloud. The integration provided by third order processes gives the researcher a combined observer and system recorded view of the session.

In Whittaker and Schwarz's paper on scheduling mediums **[Whittaker and Schwarz, 1999]**, they identify the important functions that schedules serve. Schedules serve as a joint to-do list, allowing people to coordinate future action. They can be seen as a type of contract of the work promised to be executed. Schedules also provide information without the overhead of interrupting other team members or calling a group meeting, serve as an external communication tool to people outside the group, and can assist individuals in organizing their own work.

Breakdown analysis is another more general method for studying how groups encounter problems **[Hartswood and Procter, 2000]**. The framework they propose for analysing breakdowns and repairs addresses two specific dimensions of breakdowns which may provide a basis for estimating their repair costs and can be used to derive relevant design guidelines:

- The activity context of the breakdown’s origin relative to the activity context of its detection;
- The activity context of the breakdown’s repair relative to the activity context of its detection.

The authors have derived four design guidelines for breakdown-repair scenarios.

- Choose representations and forms of artefact that afford repair;
- Support breakdown pre-emption by affording recipient design;
- Preserve the contextual information necessary to effect repair;
- Consider the actual organisation of work and its division of labour.

These sources are useful for understanding how to implement specific methods, but it is difficult to situate any given method in the larger evaluation approach or to understand how to come up with a comprehensive set of measures for addressing all of the constructs of interest to the evaluator.

Diametrically opposite to **[Hartswood and Procter, 2000]** if we consider them from the step in which the evaluation takes place, Suh et al **[Suh et al, 2007]** propose an evaluation framework to evaluate design concepts of a new product at the conceptual design phase based on users' requirements and tasks and development trends of relevant technologies. The proposed framework consists of three phases as shown in Figure 15.

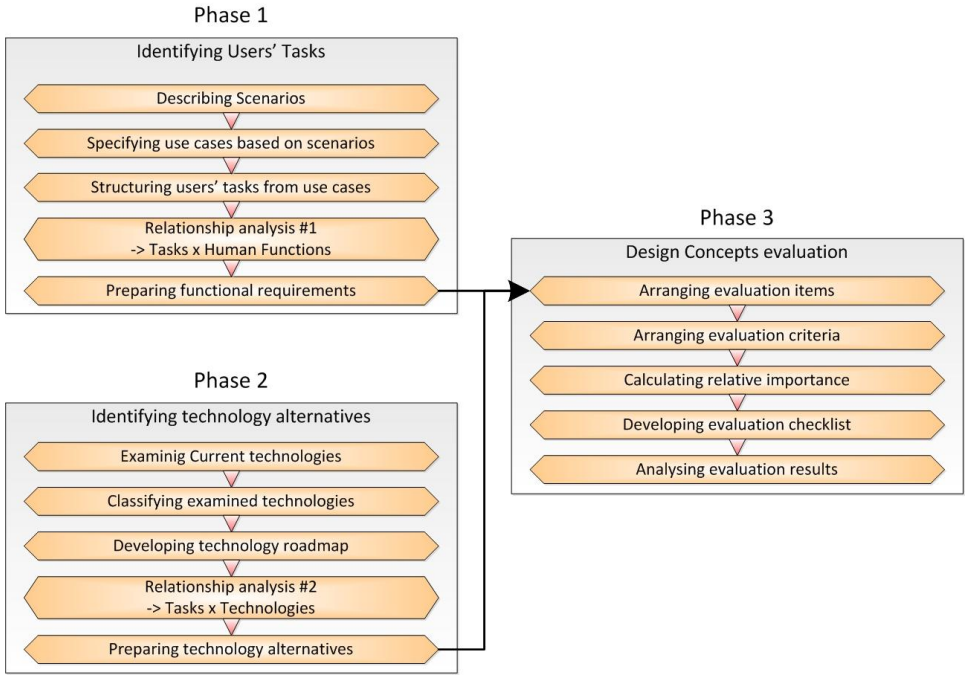

Figure 15 Framework for Evaluation of Design Concepts [Suh et al, 2007]

- *Phase 1*: identify and analyse users' needs, functional requirements and their expected tasks by utilizing user scenario-based analysis and hierarchical task analysis.
- *Phase 2*: investigate technology alternatives for satisfying the user needs or functional requirements by deploying a relevant technology roadmap.
- *Phase 3*: evaluate the design concepts using evaluation checklist, which is based on functional requirements derived from relationships analysis, utilizing CPV attribute for a quantifiable measure.

It is, however, difficult to determine how methods can be combined to form comprehensive approaches. There has been some effort to unify these differing perspectives **[Thomas, 1996]**, but much more work is needed.

As we have seen in this section conceptual frameworks and concept-oriented frameworks represents different points of view of the evaluation of CSCW systems. Conceptual frameworks rely on the description of factors to be analysed in order to evaluate a CSCW system, while concept-oriented ones focuses on some precise point of collaboration and collaborative systems. Consequently we can say that conceptual frameworks offer a higher abstraction of the CSCW domain while concept-oriented frameworks provide a more concrete approach and evaluation of systems. Thus conceptual and concept-oriented frameworks are not intended for the same use. Concept-oriented can be really powerful and efficient when you want to evaluate continuously an evolving prototype. Conceptual frameworks should be better used to evaluate a final system or to drive the development process according to some specific needs and constraints.

3.4. Synthesis

Many different techniques have been used to evaluate groupware technologies, applying approaches that range from engineering to the social kind. Such methodological variety is due to the CSCW field, and until now no consensus has been reached on which methods are appropriate in which context. Trying to apply conventional evaluation techniques to groupware applications without adapting them can be impossible or lead to dubious results. Applying the method of cognitive walkthroughs to the evaluation groupware, modifying

traditional techniques in order to be able to apply them to the study of groupware applications can be complicated and expensive.

Many researchers believe that groupware can only be evaluated by studying real collaborators in their real contexts, a process that tends to be expensive and time consuming. Others believe that it is more practical to evaluate groupware through usability inspection methods that do not make use of a real work situation.

Groupware usability evaluation is difficult to perform because the common trade-offs provided by different evaluation methods are constrained by the complex multidisciplinary nature of groupware systems. Traditionally accepted methods for assessing usability, such as laboratory experiments and field studies, become increasingly unmanageable because they involve multiple persons, which can be hard to find with the required competencies, may be geographically distributed, or simply unavailable for the considerable time necessary to accomplish collaborative tasks. Traditional experimental and laboratory methods that remove the software from its context of use may obtain simplistic results that do not generalize well to real world situations.

As an alternative to the laboratory, many groupware researches advocate the use of ethnographic and sociologic methods that explicitly consider culture and context (e.g. Quick and Dirty Ethnography). These methods have been successfully applied to real situations, but they tend to be expensive and somewhat limited. They also demand considerable time and evaluator experience. They work best at the beginning of design to articulate existing work practices and at the end to evaluate how systems already deployed in real work settings are used.

These limitations led to the emergence of a collection of discount methods: Groupware Task analysis (GTA), Collaboration Usability analysis (CUA) and Heuristic Evaluation (HE). Generally these methods lack of the capability to quantitatively predict human performance.

There are many approaches for evaluating collaboration systems, however, there are no clear guidelines or frameworks existing to support and determine what approach should be applied in order to conduct an effective evaluation. Both scientific, engineering and social science methodologies are being used, and there seems to be no agreement regarding which one has to be applied in a certain circumstance.

Another concern has been the need to develop methods that integrate evaluation and design **[Bannon, 1996]**. In order to produce better products, design and evaluation should not be seen as separate activities but as integrated parts of software development. Indeed, integrating evaluation phases into the development ensure developers, architects and even managers to develop a valid system with a higher quality. This can also save time and money by avoiding errors and misunderstanding (for example if you lead regular evaluation sessions with users you can gain in satisfaction and quality). Another interesting aspect of this integration is the fact that your evaluation will be closer from your system and then can gain in accuracy and relevance. Besides, it can give you rich feedbacks on the evaluation on the specific kind of system of yours.

As we mentioned previously **[Herskovic et al, 2007]** propose an interesting approach to CSCW evaluation by proposing a three-phased approach. This proposal relies on the principle that you don't need the same method at each step of your development. This work is one of the few we found to propose a real strategy of evaluation. It is even more valuable as it builds a frame for evaluation, meaning that instead of telling which method to use, it only gives a range of methods. Then you're free to use the best method for your system, picked-up in the right category.

Until now, no categorization or taxonomy has been able to represent evaluation methods sufficiently to provide guidelines for the complete evaluation of a system. Indeed, most of these representations only consider the qualification, the classification of evaluation

in a certain category. There is no doubt it can help designers, developers and evaluators to have a better view of evaluation methods. Still, it can only indicate to you what kind of method you should use if your consideration about the system you want to evaluate is correct. This point leverages two flaws in that kind of methodology: is your representation of the system realistic and accurate? Is this consideration sufficient to find what method or at least what kind of method to use?

Chapter 4 Evolution of CSCW

Modern Computer Supported Collaborative Work systems integrate almost all classical features required for what we can call "traditional collaboration". But some researches are going beyond these classical considerations; they want to raise collaboration to unprecedented and unexpected levels to provide new tools, features and behaviours. Among these researchers, many of them are working hard on a better management of users' context. This can be done at different steps of the human-machine interaction. For example some researches focus on the context acquisition, the different manners to collect context data by the use of hardware and software detectors. Others concentrate on the context representation by the development of context models. In an upper step, people are trying to infer information basing their inference rules and models on previously developed context representation. With inferred data, some of these scientists have designed context awareness mechanisms, allowing the adaptation of applications and systems to users' context. Another way of improvement is the one of mobility. Actually, it is simpler and simpler to collaborate anywhere and anytime. This is possible by the increasing power of mobile devices but also by the efforts headed on context adaptation. The fast emerging of pervasive computing will allow workers to collaborate seamlessly. Pervasive computing is aimed at facilitating interaction between human and machines with the final goal to make it as "natural" as possible. This chapter is organized as following: as a first step we consider the mobile CSCW which brings the classical CSCW into the mobile world by introducing the simple fact that humans rarely work on a single, with a single device, on a single project, and so on. After this first evolution we consider a critical concept for more and more domains: the context-awareness. This specific addition to the computer supported collaborative work has a special interest for multiple users systems as it can offer a way to collect and access information relative to the general system but also relative to other users. To explore a bit more the context awareness in collaborative situation we present some insights on the social-awareness domain which can be particularly useful when it comes to deal with complex collaborations situations. We continue in the different context-awareness possibilities by considering the original emotional-awareness. Before apprehending the pervasive computing and its use in current CSCW systems (which is or major interest), we rapidly present what the ambient intelligence may provide for computer supported collaborative work.

4.1. Mobile CSCW

An increasing rate of users does not work only in one place, with only one device with other immobile users. More likely they work on a laptop that they use at work, at home, in trains, almost anywhere. They can also work with a smart phone with more limited resources. Furthermore, persons they have to work with are mobile too using as many various devices as they are. Users are also "virtually mobile", meaning that their "network location" is changing as they are moving, switching to another Wi-Fi hotspot or connecting to a private network. Recent advances in wireless communications and mobile devices have made mobile handsets, such as smart phones, PDAs and laptops, very useful and popular business tools for the executives on the move. Researchers and developers have recognized the potential wireless networking can have on existing business processes and the opportunity to improve the services. Last few years, those technologies have had special attention to build mobile collaborative systems. Mobile CSCW includes **[Wiberg and Grönlund, 2000]**: 1- Individual workers move among several locations, as required by the objects of activity being to at least some part located outside of the computer; 2- People working together may be physically separated.

Let's have a look at some Middlewares supporting the mobility of co-workers:

ActiveCampus **[Griswold et al, 2003]** is a large project which provides an infra-structure focusing on integration of location-based services for academic communities. It employs a centralized and extensible architecture with five layers shown on Figure 16:

- *Data*: This layer provides for efficient storage and retrieval of data, through an SQL *database*.
- *Entity Modelling*: represents entities (e.g., users and sensors) in several forms: raw external representations (e.g., a dynamically assigned network address), fast-indexing normal forms (e.g., a permanently assigned unique integer), and forms for presentation (e.g., a screen name). It also represents the static relationships among them.
- *Situation Modelling*: synthesizes the situation of an individual entity from multiple available data sources.
- *Environment Proxy*: The Proxy layer marshals data between Devices and ActiveCampus's internals.
- *Device*: In the top Device layer, components run on devices and connect to ActiveCampus through RPC.

It supports a clear separation of the collection, the interpretation, the association with physical entities and the service-specific representation of context information. Currently, they implemented and deployed two applications: ActiveCampus Explorer, which uses students' locations to help engage them in campus life; and ActiveClass, a client-server application for enhancing participation in the classroom setting via PDAs.

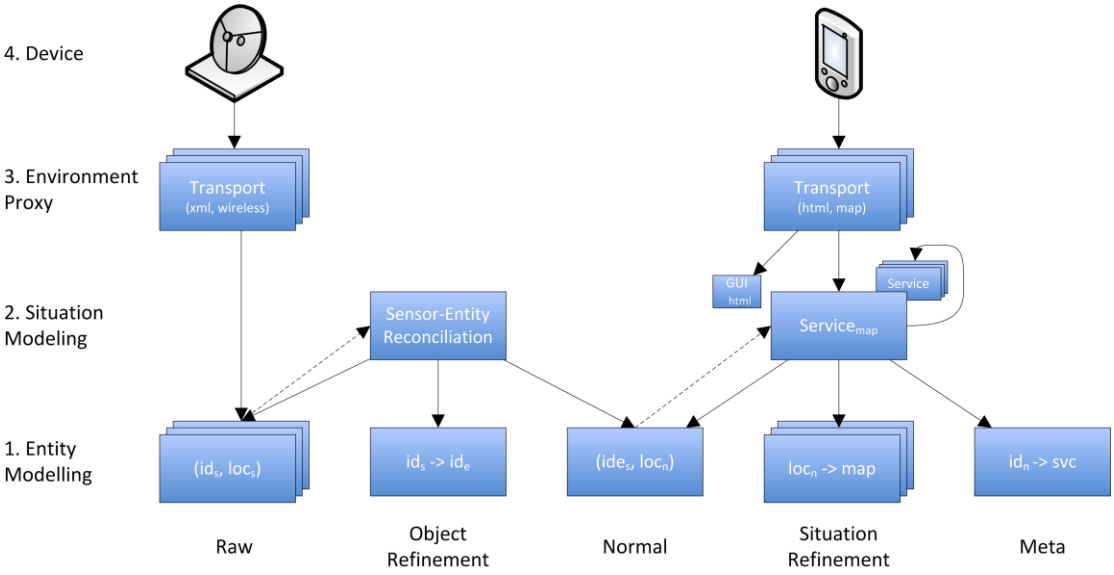

Figure 16 ActiveCampus software architecture

STEAM **[Meier and Cahil, 2003]** is an event-based middleware for collaborative applications where location plays a central role. It is a system specially designed for ad-hoc mobile networks, and hence inherently distributed. It supports filtering of event notifications both based on subject and on proximity.

YCab **[Buszko et al, 2001]** is a framework for development of collaborative services for ad-hoc networks. The framework supports asynchronous and multicast communication. The architecture includes a module for message routing and modules managing the communication, the client component and its state. Among the offered collaboration services, there is a chat, a shared white-board, and sharing of images (video-conferencing) and user files.

MoCA (Mobile Collaboration Architecture) **[Sacramento et al, 2004]** is a middleware architecture for developing context-processing services and context-sensitive

applications for mobile collaboration. The elements that compose the MoCa application are: a server, a proxy, and clients. While the first two elements execute on a wired network, the clients run on mobile devices. A proxy mediates communication between the application server and its clients. The internal MoCa infrastructure is illustrated on Figure 17. MoCa provides three main services: Context Information Service (CIS), Symbolic Region Manager (SRM) and Location Inference Service (LIS). CIS receives and processes contextual information sent by clients. It also receives notification requests from the application proxies and delivers events to whenever a change in a client's state is of interest to this proxy. SRM offers an interface to define and request information about hierarchies of symbolic regions that are meaningful to location-aware applications. Based on SRM information, LIS infers the location of a mobile device from the raw context information collected by CIS of this device. The communication substrate consists of the publish-subscribe mechanism and a communication protocol. The former provides the basic functionality to the CIS, once the context recognition is done by the definition of subscriptions that specify a set of user-defined context information.

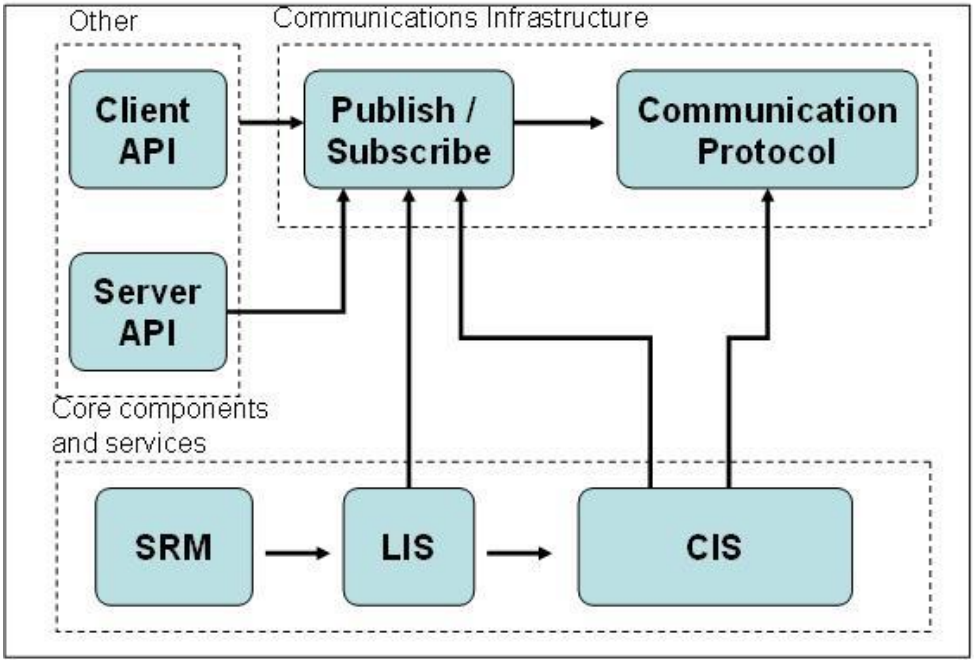

Figure 17 MoCA internal infrastructure

Sync **[Munson and Dewan, 1997]** is a Java-based framework for developing collaborative applications for wireless mobile systems. Sync is based on object-oriented replication and offers high-level synchronization-aware classes based on existing Java classes. Sync allows asynchronous collaboration between mobile users. Sync provides collaboration based on shared objects which can be derived from a Java library. As in Bayou (see below), data conflicts are handled by the application. Sync applications have to provide a merge matrix, which contains a resulting operation for each pair of possible conflicting operations. With the help of the merge matrix, conflicts can be resolved automatically.

Bayou **[Terry et al, 1995]** provides data distribution with the help of a number of servers, thus segmented networks can be handled. Bayou applications have to provide conflict detection and resolution mechanism, thus no user intervention is necessary. Bayou is not designed to support real-time applications. Bayou is a replicated weakly consistent storage system designed for a mobile computing environment that includes portables machines with less than ideal network connectivity. To maximize availability, users can read and write any accessible replica. Bayou's design has focused on supporting application-specific mechanisms to detect and resolve the update conflicts that naturally arise in such a system, ensuring that replicas move towards eventual consistency, and defining a protocol by which the resolution

of update conflicts stabilizes. It includes novel methods for conflict detection, called dependency checks, and per-write conflict resolution based on client-provided merge procedures. To guarantee eventual consistency, Bayou servers must be able to roll back the effects of previously executed writes and redo them according to a global serialization order. Furthermore, Bayou permits clients to observe the results of all writes received by a server, including tentative writes whose conflicts have not been ultimately resolved.

Quickstep **[Roth and Unger, 2001]** is a platform specially designed for groupware applications, 'Quickstep' running on handheld devices. Group Management, managing data, mirroring and caching have been integrated in this system. However, due to the inefficient communication infrastructure (using Bluetooth, IrDA, and Serial Connection), it is not very effective in real collaborative environment. The QuickStep platform is designed to develop both, collaboration aware and mobility aware applications. It provides awareness widgets for collaboration awareness as well as for context awareness. The QuickStep approach can be described as follows: QuickStep supports applications with well-structured, record oriented data, as being used by built-in software for handheld devices (e.g. for to-do lists, memos, telephone lists). It has explicitly not been designed for supporting multimedia data, graphical oriented applications or continuous data streams and is mainly designed for supporting synchronous collaboration. It provides awareness widgets for collaboration awareness as well as context awareness. QuickStep comes along with a generic server application which allows supporting arbitrary client applications without modifying or reconfiguring the server. The system also ensures privacy of individual data.

Dustdar **[Dustdar and Gall, 2003]** describes a framework for process-aware distributed and mobile teamwork. It decomposes process and workspace management issues and presenting a three layered architecture, which integrates process awareness with the easy to use groupware (workspace) metaphor. A three-layer architecture that integrates workspace management, publish-subscribe, and community/user management. The system has been implemented and tested on a peer-to-peer middleware. Moreover a three-layer software architecture for distributed and mobile collaborative (DMC) systems, which provides mobility of context to its group members, is proposed. This architecture defines a foundation for the flexible integration of Collaborative System (such as Workflow Management, Groupware or business Process Modeling) with teamwork services that support distributed and mobile collaboration.

YACO (Yet Another Framework for Supporting Mobile and Collaborative Work) **[Caporuscio and Inveraldi, 2003]** is a CSCW framework that aims at supporting mobile users. It is based on the publish/subscribe paradigm and apply it to mobility. YACO uses two already existing systems: SIENA **[Carzaniga et al, 2000]** which a publish/subscribe system known to have good scalability and MobiKit **[Caporuscio et al, 2003]** which is a service for mobile/subscribe applications. The system is based on the clients/servers architecture of SIENA. Clients are the final users of the framework but also some special clients such as databases or else while servers are some kind of messages routers. In addition to these two entities, the systems rely on another kind: the mobility proxies which manage the publish/subscribe aspect for a given set of clients. When a client goes offline, the proxy that was associated with it stores the messages that are sent to the client. When the client comes back online and connects to another proxy, the first one forwards the stored messages back to him. On that way, no messages are lost during the offline time. In addition to this, YACO defines two kinds of profile: user profile and document profile, which are XML definitions. The user profile gives some information such as name, group memberships, and knowledge. The document profile gives classical information about a specific shared document (author, permissions, description, title, etc.). YACO allows users to exchange messages and files. This work is interesting as it allows a user to be mobile and keep its collaborative ability. Besides, the publish/subscribe paradigm seems to be really effective for mobile collaboration as it saves mobile device resources and "filters" information to communicate only relevant data.

Now that we have travelled among some of the frameworks dedicated to the mobile CSCW, let's have a look at the kind of applications they may have to support and which have already been proposed.

StudySpace **[schnase et al, 1995]** attacks problems such as determining network, hardware and display capabilities before fetching a document; it does not address subscription, distributed searching and community support issues.

Dacia **[Litiu and Prakash, 2000]** is a system that provides mechanisms for building groupware applications that adapt to available resources. Using DACIA, components of a groupware application can be moved to different hosts during execution, while maintaining communication connectivity with groupware services and other users. The system however, does not provide higher-level service support for notification and information sharing. Important mobile teamwork features such as file sharing, user awareness (i.e. notification that user X is online) and access control are not addressed by DACIA.

Most **[Cheverst et al, 1999]** provides five components: a group coordination module, a shared graphical editor module, a remote database module, a collaborative viewing module and a job dispatching module. McKinley and Li **[McKinley and Li, 1999]** proposed a synchronous collaboration application for handheld computers called 'Pocket Pavilion'. They have employed a component based Java framework which is capable of reusing existing components or enabling users to plug-in other helper applications. The framework utilizes a 'Leader-Followers' environment where their collaborative session is monitored by the leader and the session is multi-casted to the followers using an extensible leadership protocol. Pocket Pavilion is a web based application that extends collaborative browsing to wireless handheld computing platforms running Windows CE.

MOTION **[Kirda et al, 2002]** (Mobile Teamwork Infrastructure for Organization Networking) is oriented to manage user's mobility and is based on a peer-to-peer architecture. The system is composed of peers that communicate with each other by using services offered by a middleware called PeerWare. Services offered by MOTION are: artefact storing in local repository, resource searching and sharing, messaging, system events subscribing. The MOTION service architecture supports mobile teamwork by taking into account the different connectivity modes of users, provides access support for various devices such as laptop and mobile phones, and uses XML meta-data and the XML Query Language (XQL) for distributed searches and subscriptions. Besides, MOTION offers a full API of TeamWork Services (TWS) allowing anyone to build business specific applications onto the system. The MOTION system is composed of three layers as shown in Figure 18. The communication layer, the teamwork services layer and the presentation layer.

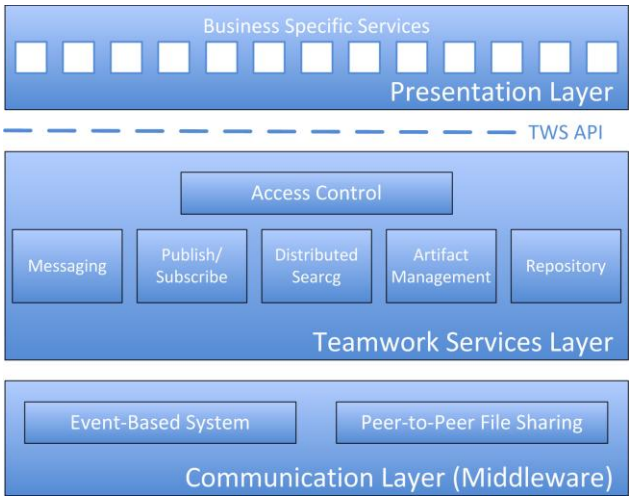

Figure 18 The layered MOTION architecture

- ***Communication layer***: provides an event-based infrastructure for publish/subscribe and peer-to-peer distributed file sharing support. It provides low-level distributed search propagation and invocation services to the upper layers and an API that allows the upper layers to subscribe to and get notified of events generated by system components.
- ***Teamwork services layer (TWS)***: is responsible for the integration of the main components of the system (e.g., access control, user and community management, repository). The TWS layer provides an API to generic services such as storing and retrieving artefacts in the local repository and from remote repositories on other peers, creating and managing virtual communities, sending and receiving messages from other users and distributed search specification and invocation.
- ***Presentation layer***: provides the user interface to the MOTION services and is built using the TWS API.

Kirda et al. **[Kirda et al, 2003]** have developed service architecture for mobile teamwork. The system takes into account the different connectivity modes of users and uses XML meta-data and XML Query Language (XQL) for distributed document searches and subscriptions.

Su et al **[Su et al, 2004]** proposes a multimedia mobile collaborative system based on content-aware, device-aware and connection-aware framework. In this framework there are four major components shown on Figure 19.

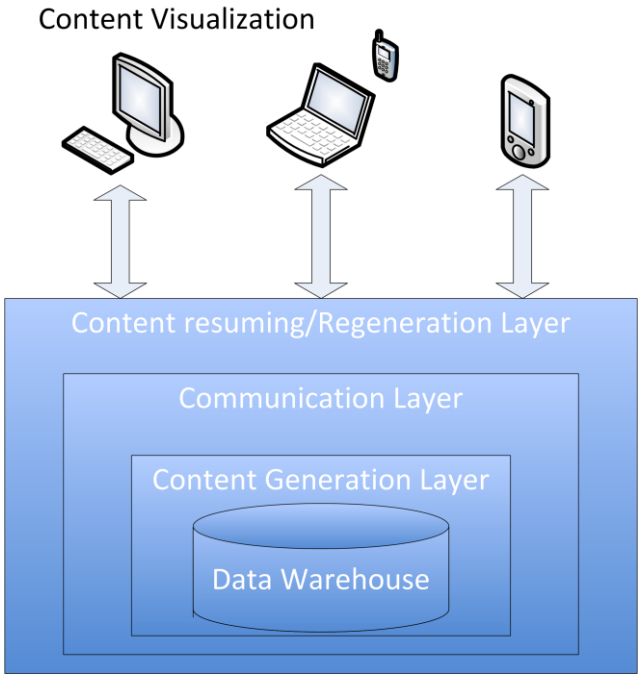

Figure 19 Framework of mobile collaborative environment [Su et al, 2004]

- *Content Generation Layer*: in this layer, the content server generates the unified content based on client request;
- *Communication Layer*: maintains each interaction session and delivers messages between client and server;
- *Content Consuming and Regeneration Layer*: messages are assembled and sent to content visualization layer;
- *Content Visualisation Layer*: a content viewer is used to display each object.

RoamWare **[Wiberg, 2001]** has been developed as a mobile CSCW physical/virtual meeting support system intended to support knowledge management (KM) in mobile CSCW. This system takes into account the situated nature of meetings taking place at several different

locations apart from ordinary meeting rooms. The RoamWare system consists of the three subsystems RoamList, RoamWeb and RoamLines. These three subsystems support three main activities which are: RoamList: Recording meetings, RoamWeb: sharing meetings, RoamLines: searching meetings.

[Lee et al, 2004] proposed an agent-based computer supported collaborative work system known as TeamWorker that enables collaboration between mobile workers for the execution of their jobs by providing services via conversational components (C-COMs). The main component of TeamWorker is a smart agent, called a Personal Agent, which resides on a mobile computing device and plays a personal assistant role for a mobile worker. The benefits of TeamWorker are: improved cooperation among mobile workers, increased reachability of remote workers by supporting multi-modal interactions between them and their personal agents, and better workforce management via transparent work progress monitoring.

It is anticipated that a mobile collaborative system will play a very important role in many enterprise activities. Enterprise mobility over wireless networks will need a robust infrastructure for information exchange between different business and enterprise entities. However, existing mobile collaborative systems are still not mature enough to meet this need. There are a number of issues, which make the implementation of a mobile collaborative system very challenging. Firstly, on device side, the screen is small; lacks sufficient computational power (despite recent improvements); short battery life; limited storage space, etc. Secondly, on the connection side, regardless of the communication protocol used (e.g. wide area wireless communication protocols such as GSM, CDMA, and UMTS), wireless connections are unreliable and provide modest bandwidth. Therefore, direct migration from traditional Collaborative System to Mobile Collaborative System will be difficult and will not provide the same degree of satisfaction and sophistication to the users.

Collaborative as well as mobile applications have to keep data consistent. Applications which are collaborative and mobile at the same time square the problem of data consistency. Collaborative applications have to synchronise concurrent data manipulations, mobile applications have to keep data consistent when devices are moved inside the network or are disconnected from the network. Even if new problems emerge from the introduction of mobile computing in the computer supported collaborative work, this new aspect also offers unprecedented opportunities for co-workers to keep their workflow alive while moving to another location or if they have to change the device they work with. Mobile computing also extends the context of the collaboration, indeed, as users become mobile (physically and logically), they multiply the dimensions required to correctly represent their context. For instance, if location wasn't a real problem when everyone only had a single workstation at their office, now that they have a workstation, a laptop for business travel, a professional cell-phone, as well as the possibility to simply connect on an hotel's computer and securely access to their professional network, the representation of context and its adaptation has become a crucial need for mobile CSCW.

4.2. Context-awareness CSCW

Several definitions of awareness exist in the CSCW literature, which are not always mutually exclusive. Presence awareness, for example, refers to the knowledge of who is around **[Milewski and Smith]**, while workspace awareness refers to the knowledge of each other's activities within a shared work context **[Gutwin and Greenberg, 2002]**. Peripheral awareness **[Grudin, 2001]**, in turn, refers to one's ability to capture snapshots of the others' activities while focusing on a different task. The term awareness refers to actors' taking heed of the context of their joint effort, to a person being or becoming aware of something **[Schmidt, 2002].** There also exist definitions for contextual awareness **[Izadi et al, 2002]**, passive awareness **[Dourish and Belloti, 1992]** Dourish and Bellotti define awareness as discernment of activities performed by other people, provided by information about their context. Without context knowledge, participants do not know what the others are doing,

resulting in rework and/or task inconsistencies, as well as preventing spontaneous communications to be initiated. Through contextualization, important contributions can be provided and assessment can be made, aiming, above all at reaching pre-established goals as well as being informed about the progress of the group, and situation awareness **[Gutwin and Greenberg, 2002]**. Kyung-Kim and Kim identify five important types of awareness required for group collaboration and communication. **[Kyung-Kim and Kim, 2006]**: task awareness, member awareness, presence awareness, schedule awareness and activity awareness.

Awareness is an integral CSCW research component. The core challenge for CSCW systems is providing effective support for activity awareness. Collaborators who are not able to be at the same place at the same time need continuous support to remain aware of the presence of their counterparts, their tools and other resources, their knowledge and expectations, their persistent attitudes and current goals, the criteria they will use to evaluate joint outcomes, and the current focus of their attention and action. Many concepts of awareness have been discussed in CSCW literature: social, presence, action, workspace, situation awareness. This variety is itself an indication of the importance of awareness for CSCW designs, and of course to the experiences of users.

A collaboration-aware application is one that was developed with the aim of allowing multiple users to work cooperatively. Liccardi et al proposed CAWS **[Liccardi et al, 2007]** a collaborative authoring system specifically designed to improve awareness in users. It is developed around the wiki concept. It aims to provide a tool for writing professional papers for conferences and journals.

According to **[Streitz, 2001]** the following areas have to be integrated into an umbrella framework for future CSCW research: Pervasive computing, the disappearing computer in combination with augmented reality, architecture and hybrid worlds, and new forms of human-computer interaction.

As we have seen, the context-awareness can take various forms and apply to different aspects of the context. If we had wanted to give a general definition of context-awareness we could have said that it simply corresponds to the ability to stay aware of any information that can be relevant for the past, current and future activity. Context awareness for CSCW systems are most of the time more complex than for single user environments. As it has been evoked, the number of dimensions required toward the complete description of the context is multiplied by the simple fact that the collaboration itself is a complex object to represent. Probably more than with any other aspect of the context, the collaboration between humans generates complex relations among them which can dramatically influence and alter their work. Obviously, these social relations can be hard to describe, but as they are relevant to the collaboration activity, they should be integrated into the representation of the context. In the following section we will present some works toward the correct representation and use of this social awareness.

4.3. Social Awareness

The social aspect in computer sciences is a popular domain which has known a recent growth with the help of social networking websites such as MySpace[11] or Facebook[12]. The term “social” could be defined as the relationship between a group of persons. Meaning that to create a social network you just have to put two persons in relation. However, what we can call *social context* handling is one the major challenges of modern computer sciences. Indeed, the social relationships are particularly hard to represent and complex to understand, even for humans. Despite this inherent complexity, some aspects of this domain can be managed and used by computers. An interesting point of this domain is the one of the *social context*

[11] http://www.myspace.com/

[12] http://www.facebook.com/

awareness of user. Explicitly it is the ability of a computer to present to the user a part of his social context, and thus a part of his acquaintances' context.

Jakob E. Bardram and Thomas R. Hansen **[Bardram and Hansen, 2004]** present an interesting article on an architecture to support social awareness in mobile cooperation. They base their researches on the fact that co-located workers have access to a part of the context of each-other while non co-located workers miss a part of it, by losing sight contact. All the article is based on a specific case on a hospital were nurses, young doctors and experienced ones must collaborate in order to benefit from each other and work in an efficient manner. In order to tackle this problem, the authors present a mechanism called *Context-Mediated Social Awareness*. To support this mechanism they also propose a general-purpose architecture allowing the development of future applications using social awareness. According to them, "*the concept of Context-Mediated Social Awareness aims at minimizing unwanted interruptions between mobile, distributed co-workers by enabling a social awareness through the use of context-awareness systems*". One of the main issues they want to solve is the one of interruptions. Interruptions are unexpected or inappropriate breaks in the flow of work. They are often caused by a co-worker wanting to have some information. They report "*90% of brief conversations are unplanned, and hence are potentially interruptive*", moreover '*only 55% of people who are interrupted continue their previous activity*". Furthermore they report that ethnographic studies show that co-located workers tacitly coordinate their work activities and communications avoiding to interrupt each other. Social awareness can then be characterized as the ability to integrate the context of others in order to adapt his own behaviour. This awareness can be described by two aspects, namely "monitoring" and "displaying". Monitoring is the action by which a person observes the current actions of another. Displaying is the action by which the user tries to communicate a part of his context to another. In traditional collaboration activities people use social artefacts in order to channel social awareness. In the hospital use case there are some examples of such artefacts: messages on whiteboards, use of calendars and so on. However, those artefacts are located in a specific place, whereas users are moving along different locations and then can't be aware of those artefacts. The authors "*suggest that computers can support context-mediated social awareness by presenting remote users with context cues*". To develop their application authors have developed a first prototype and led an evaluation in a hospital with some collaboration scenario. After this first phase, they designed and developed a framework, named AWARE, to support context-mediated social awareness and re-implemented their first prototype with some changes based on users comments. In addition to this architecture, they have developed a mobile phone application over the Symbian[13] operating system to use the AWARE framework and provide social awareness to users.

The AWARE framework, by relying on a context-awareness infrastructure, proposes new ways of handling social awareness. By basing their work on a preliminary evaluation they also have respected wishes of users. Finally, as the framework is aimed for general purpose it can be adapted and enriched for specific cases. The social-awareness, despite its undeniable interest is maybe one of the most complicated aspects of the context awareness; it may imply the measurement of numerous parameters but also the estimation of some of them. The AWARE framework is interesting in the sense that it takes into consideration the current occupation of a worker, however it may fail to address some other important aspects of the social awareness such as the precise "evaluation" of people's relations. In this perspective, there is a specific point, extremely hard to handle which has been left untreated, probably due to its complexity and the fact that it may require considerable resources to be considered efficiently: the evaluation of emotions and its use, the emotional awareness.

[13] http://www.symbian.com/index.asp

4.4. Emotional awareness

One of the most important characteristic of groupware systems when compared with other multi-user systems is their ability to convey information of other participant's presence and their role during collaboration. This is referred to as Collaboration Awareness. Even if the consideration of roles, tasks and other classical collaboration aspects is critical for awareness mechanisms, it is important not to forget that the collaboration takes place between humans. In this perspective, it can be essential to keep and channel the awareness of "less collaborative" aspects. Given this consideration, Garcia **[Garcia et al, 2000]** proposes a type of collaboration awareness, namely Emotional Awareness. There are numerous ways in which emotional awareness can be used in groupware.

- Groupware conceived for particular activities of a meeting board could benefit by incorporating emotional awareness **[Garcia et al, 2000]** as an additional collaboration and communication aid.
- Another application of emotional awareness in CSCW can be found in collaborative learning
- Other important application of emotional awareness exists in information retrieval in organizational memory systems, by using perceptual information retrieval and as implicit feedback input for recommendation systems.

Defining, describing and managing various and exotic types of awareness are essential parts of the collaboration awareness domain. However one has to keep in mind that such awareness must be used for greater purposes. Pervasive Computing and Ambient Intelligence could be some of these.

4.5. Ambient Intelligence

Ambient intelligence is a recent domain of electronic and computing, it can be defined as an evolution toward the omnipresence and full orchestration of devices of the environment. One of the principles of ambient intelligence is the fact the environment is "filled" with electronic devices and sensors, permanently capturing and analysing this environment. From capture, analysis and communication of sensed data, electronic devices can became almost completely context-aware and thus, given the fact that they have proper reasoning mechanisms, optimally adapt their behaviour. **[Weber et al, 2005]** mention additional characteristics of ambient intelligence such as that it can be "enabled by simple and effortless interactions, attuned to all our senses, adaptive to users and context-sensitive and autonomous." Ambient intelligence requires that the environment is embedded, adapts to the presence of people and objects and assists users smartly while preserving security and privacy. And ambient intelligence supports human contacts **[Weber et al, 2005]**. Benefits that could be gained by CSCW with the help of ambient intelligence are numerous; among them we can point out the collection of data, helping to have better context awareness, the adapted behaviour of devices according to this context and the potential evolution of collaborative habits due to this intelligence.

Cooperative ambient intelligence (CAI) **[Gross, 2008]** is based on ambient intelligence that "aims to improve users" work and private life by analysing and adapting to the current situation with a special focus on the presence and activities of users" **[Fetter and Gross, 2007]**, thus cooperative ambient intelligence is currently emerging. The author summarizes the evolution of users' interactions and systems autonomy and adaptive behaviour from the 1970s on Tab 8 . As we can see, the most recent evolution is the CAI. Finally they point out five main research issues for the CAI:

- *Dynamic user and group authorisation and authentication;*
- *Dynamic access rights;*

- *Advanced context models and meta-models;*
- *Machine learning;*
- *Advanced hardware and network technology;*

	Single-user WIMP	*Cooperative Systems*	*Single-user Unicom*	*Single-user Ambient*	*Cooperative Ambient*
Time	1970s	1980s	1990s	1990s	2000s
Key Concepts	Graphical user interfaces; icons, menus and pointing devices	Graphical user interfaces; computer-based communication screen sharing and telepointers	Vast availability or embedded and mobile technology periphery of user attention	Attuned to users' senses; adaptivity part of any environment.	Adaptivity to groups of users part of any environment
Autonomous and adaptive behaviour	Adaptability through user customisation; adaptivity of user interface	Adaptability through articulation works support; adaptivity of resource allocation	Adaptivity of information and functionality	Highly adaptive to (mostly single users') presence and activities	Highly adaptive to presence and activities of local and remote groups, and to their needs for local group interaction and remote group-group interaction

Tab 8 Evolution of interactions for [Gross 2008]

Ambient Intelligence is a highly promising domain and it will surely be unavoidable in some years. However, our point of view is that given the fact that ambient intelligence is built over the Pervasive Computing paradigm, the first step to be able to properly consider ambient intelligence is to correctly and robustly integrate Pervasive Computing within CSCW systems. Once it will be properly done, the evolution toward complex ambient intelligence should be natural.

4.6. Pervasive Computing

Pervasive computing refers to a vision of future computing environments first described by Mark Weiser in his 1991 influential paper "*The Computer for the 21st Century*" **[Weiser, 1991]**. In a general way, we can say that Weiser envisioned a world where computing is embedded in every-day objects that interact with each other to perform actions on behalf of the user. Today's vision of the Pervasive Computing is close to this definition. Indeed, pervasive computing is now considered as the miniaturisation and multiplication of smarter and smarter devices, allowing companies to embed electronics in almost any kind of object. In addition, the Pervasive Computing paradigm advocates for the extensive connectivity of these devices, allowing them to intensively communicate. Another essential element of the Pervasive Computing is the capacity of devices to share information and adapt their behaviour according to those. Thus, we could summarize the Pervasive Computing as a will to make any environment information available to any device, in order for them be able to adapt their behaviour as accurately as possible and then finally optimize user's experience.

One of the main aims of pervasive computing is to be invisible for the users. For that, applications embedded in electronic devices must be proactive, by "guessing" the needs of the users and providing them with relevant context information on the right place and at the right time. Whereas some context information can be obtained from the local device, such as the agenda of the user, others can only be acquired by interacting with nearby hosts. For such interaction to take place, a device must first discover, or sense, the hosts around it.

A natural extension for CSCW is then to be more suited for this kind of environment where the number and the type of devices is nothing more than a variable to which you have to be able to adapt. In this perspective, an interesting challenge for CSCW is to know dynamically and efficiently how to take advantages of this technology to improve the collaboration between humans. Potential advantages are various and virtually unlimited, we may think to simple mechanisms such as smart device control via a computer until complex mechanisms of collaborative context awareness.

Among the few works that we found toward the consideration of pervasive computing within collaborative work there is one that particularly dragged our attention. Thus, in **[Reiff-Marganiec et al, 2008]**, the authors propose an architecture designed to dynamically compose pervasive services in a collaboration context: the PCSA (Pervasive Collaboration Service Architecture). In this interesting work, they start from the assumption that even if more and more types of pervasive services and collaboration services are developed, there is no real try to organize them or to make them work together and interoperate. Standing from this point, the PCSA has for objective to help the interoperability of these services and make it transparent for users. As it has been presented by authors, the architecture is composed of three main elements: the user applications, the collaboration services and the inContext core platform. There's not much to say about user applications, except that they are custom built applications to respond to user's need by using collaboration services. These services, as the second element of the architecture, are supposed to be a variable set of tools enabling the collaboration between users. Finally, the core platform is subdivided into four parts: the Access Layer controlling and logging any access to the platform; the context management in charge of dealing with context information providing and enriching; the service management dedicated to the registry, lookup and call to the collaboration services and finally the interaction mining element analysing behaviours of users and services to enrich future interactions and services adaptation. There's no doubt that this work is interesting, indeed it addresses the pervasive computing and collaboration domains from a service composition perspective. However, if we compare to what we tend to realize, there are some obvious and hardly overcoming issues.

Our first concern about this work is that it forces the development of new tools to ensure the collaboration. Indeed, in the example proposed, they quickly depict an application they had to build to correctly address the need of their partner. Even if this new tool is effective, it will force end users to handle a new application. This may not seem to be a great problem for a single tool, but it can surely become a serious issue if you consider complex collaborative environments such as Lotus Notes.

Another concern about this work is the lack of pro-activeness of users' applications. Indeed, even if they are able to compose different services according to the current context and try to evaluate the different benefits of each service to make the best composition, they do not address the simple fact that "pervasive devices" has to anticipate their future context and do not simply respond to this context, but effectively try to modulate it in order to optimize the future situation.

To conclude on this work, we may say that even if the underlying idea of helping the interoperability and composition of services is doubtlessly a good one, the concrete architecture they have built does not succeed in fulfilling some of basic pervasive applications requirements we want to incorporate in our work.

4.7 Synthesis

The first attempt to use computers for collaboration is now forty years old while the first collaborative software is twenty. The computer supported collaborative work domain is probably one of the most active research fields of recent years. Indeed, due to the facilitations brought by computers and smart devices it is almost impossible to find people working without them. For the past few years information technologies have been evolving toward the multiplication of smart electronic devices such as smartphones, laptops, GPS and so on. Despite or maybe because of this proliferation the digital environment is a non-continuous space where miscellaneous devices can communicate, or not, with others. Thus, in order to make this space "continuous" the Pervasive Computing is based on the communication between devices to smartly adapt their behaviour to the current context of users and offer them a seamless interaction with the digital world. Given this aspect our work has rapidly focused on the way we could integrate the pervasive computing within CSCW. Such integration could bring various advantages: resource and time saving for companies, work simplification and task automation for workers. In a "green" consideration it could also help reducing work's energetic impact by accompanying users' in using lighter devices and services.

However, only few advances have been made toward the better integration of pervasive features in CSCW systems. Thus the main objective of our work was to provide a model to natively support the pervasive computing paradigm inside the collaboration of users. This model has to help improve the continuity of service among the different devices of the environment, but also try to anticipate possible discontinuities according to the collaborative context of users. As we will see in the next sections this model relies on several sub models: a role model, a task model and a resource model. Those models have in charge to represent the context of users implied in the collaboration along with the different aspects of this collaboration and resources required to complete the different tasks and objectives of the collaboration. Naturally they're also used to represent the current state of the context, as it will be used to find adapted behaviour for devices. Thus, those models are coupled with the definition of "device collaboration rules" that provide patterns of collaboration between devices according to generic states of context. As it will be fully explained in the dedicated sections, the context model is naturally completed by the set of rules and their association enable a seamless and efficient collaboration of devices based on the needs expressed by the collaboration of humans.

Part 2: Extending Collaborative Work through Pervasive Computing

Chapter 5. Pervasive surveying in situation of mobility

Chapter 6. PCSCW model

Chapter 7. Evaluation of Pervasive CSCW

Chapter 8. Simulator

Chapter 9. Evaluation and Validation of PCSCW Model

Chapter 5 Pervasive Surveying in situation of Mobility

Our research interest has focused on the integration of the pervasive computing aspect in the computer supported collaborative work. On the long road toward this accomplishment, our first step was to propose a new service coupling both concepts: the Pervasive Surveying. The goal of P-Surveying is to support an enhanced and ergonomic cooperation between mobile users. One of the main ideas brought by this concept is that the initiator of the cooperation facilitates the contribution of other members of the group. This is allowed, notably; thanks to the use of semi open multiple choices questionnaires (SOMCQ). In addition, this service will be used in the following chapters as a base to illustrate our model.

5.1 Use case

Let's consider a use case involving a manager with his distant and spread working team. The manager is visiting a road show presenting new devices. He wants to have as soon as possible the main opinion of his team on the interest of the device for the company. Without specific system for collaborative work, he will have to try to contact directly each member of his team, by a phone call, a SMS or a mail. In the case where the team has his own forum, the manager will leave a message, eventually with a questionnaire. Then, if his team is composed of a great number of members, he will certainly be forced to use paper and a pen to sum up the votes (in the case where there are not using a forum). In the other case, we will have to wait that member's check the forum for new messages. We can point out several lacks of classic means of communications in the context of collaborative work in situation of mobility:

- Contacting each member of the team can take too much time or be costly;
- Except for forum surveys, we can't summarize quickly the opinion of the group;
- Forums, in opposition to mails, sms and phone calls do not contact directly team members (asynchronous communication);
- Mobile (such as PDA and cell-phones) devices are not completely suited for comfortable web browsing.

Thus, our goal for this first approach to the coupling of pervasive and collaborative aspects was to find a way to improve this situation by providing an efficient service to help the collaboration.

5.2 Pervasive Surveying system

In **[Lancieri et al, 2005]** they propose a methodology allowing the generation and the use of online multiple-choice questionnaires to enhance collaborative work. This work is mainly based on the use of small tags such as "q-", "i-" or "r-" to generate questionnaires from a simple mail. Based on this work we propose both a collaborative methodology and a technological process that support pervasive collaboration.

The methodology starts from the idea that questionnaires are an ergonomic and effective way to collaborate in non-comfortable situation (users not in face-to-face meeting, mobility situation, etc). The automatic process has for goal to simplify the questionnaire creation and the reuse of results.

This process is based on the use of mail and sms based. It allows users, in a few words, to send messages which will be transformed in questionnaire on a collaborative website. One other major part of this user friendly process consists in making a synthesis of user collaboration and presenting it ergonomically. This is a critical need, because we clearly

know that synthesis are hard to achieve in collaborative situation and that an efficient service with an awful interface will be less used than a more classic service with a good one.

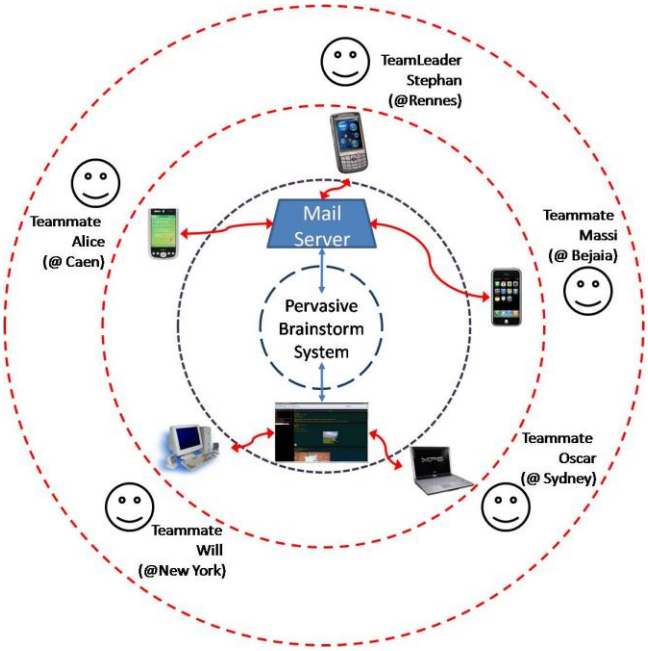

Figure 20 Pervasive Surveying System

Figure 20 represents how a team can collaborate around a simple system offering two simple interfaces: a web page and a mail box.

The main system has several functionalities:

- Collect mails from the mailbox;
- Analyse and transform these mails into html;
- Compute averages of users answers;
- Display questionnaires on the webpage as well as statistics results.

The outer layer represents the team community. As we can see, they're in different locations, do not have the same devices but have to work together. The second layer is the representation of team member's devices, they're of different kinds and offer various connectivity. The third layer is a reference to the different available networks and protocols built over them. (Internet, http, smtp, imap …). The core is the pervasive surveying system itself allowing interactions between all members of the community.

Let's go back to our team manager. He still needs to have the opinion of his team concerning a new device. As he wants to have the opinion of each member of his team, the best way to do that is to create a questionnaire with a main questions and a set of possible responses. With our system, and given the fact that he only has a smartphone, he can simply write a mail with a simple syntax and send it to a P-Surveying service dedicated mail box. Once this is done, our system automatically takes the mail, extracts the questionnaire and publishes it on the collaborative web site. Finally our system sends a mail back to the manager which informs him at what URL the questionnaire is available. The manager can then forward the mail to his team. Actually, this approach allows anyone with any terminal to collaborate with any groups. After that, each member of the team can directly access to the new questionnaire and vote and optionally propose a comment. At any moment, the manager can access to the website to consult his questionnaire results. Finally after few hours, the manager can have a synthetic formalized response from a large part of his staff, thus he can, in one eye shot take a decision that will reflect the opinion of the group. In the following part we will describe a system designed to enable those interactions.

5.2.1. Publication of questionnaire

Let's have a look at the questionnaire publication service. Figure 21 shows the life cycle of a questionnaire with our system.

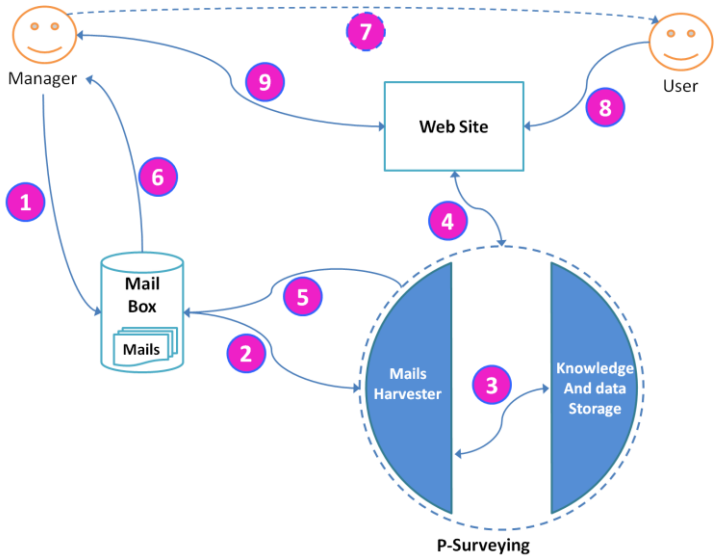

Figure 21 Questionnaire Life Cycle

1. The first step is to compose a questionnaire with a communication device and a mail client. To be sure that the questionnaire will be correctly interpreted by the service, you need to respect some simple syntax rules and use the set of predefined tags (e.g. q- for a question and r- for possible answer). As this rules are quite simple and not too numerous, we assume that someone can use it almost naturally. Once the questionnaire is composed, you have to send it to the mailbox of the p-surveying service;
2. The second step is the harvest of the mails by the system;
3. After that the system analyses the mail, extracts the questionnaire and stores it in the data storage (which will be described later);
4. The questionnaire is converted to HTML and published on the website;
5. Fifth, the system sends back a mail to you with the link to the questionnaire on the website;
6. As sixth step you have to forward the mail to all users you want to warn;
7. All users can access to the questionnaire on the site with the link contained in the forwarded mail;
8. Finally, you or any member of your team can access to the results of the questionnaire.

This relatively simple system provides a simple way for anyone to publish its own questionnaire, even if he only has access to a mail client.

5.2.2. System architecture

After this general overview, let's have a closer look at the core of the system. As it can be seen from the precedent figure, the system we had to develop required several elements to ensure the functioning: a mail box and a mail server to be able to receive and send mails to members of the team, a website to publish the questionnaires and a middleware to extract mails from the server, analyse them and allow the publication of extracted questionnaires on the website. Figure 22 shows the architecture of our system. The system itself was designed to be composed of the following parts:

- **Mail Server:** a classic mail server.
- **Mail Harvester:** a composite component in charge of collecting and fetching mails from a mail server.

- **Mail Server Interface:** the interface between the mail server and the mail harvester. We used the JavaMail API to build this component.
- **Mail Reader:** a component in charge of reading mails from our data structure..
- **Mail Saver:** the component in charge of saving mails (in the data structure and the attachments in the dedicated folder).
- **Data Structure:** a specific data structure used to save mails structure, links to attachments and questionnaire.
- **Storage Folder**: a folder to store mails attachments and textual contents.
- **Publication Web Site:** the website where mails and questionnaires are published. It is composed of a set of Servlets.
- **Questionnaire Parser:** a mail content analyser. It parses the textual content of the mail and extracts questionnaires from it.

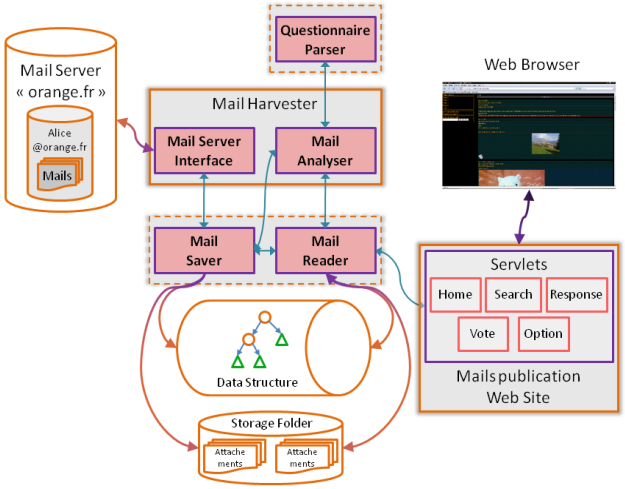

Figure 22 System architecture

During the conception of the system, we had planned to use an ontology to represent and store messages (mails and questionnaires) exchanged between the different survey participants. We initially wanted to use this specific representation in order to be able to analyse and reason more easily on the different messages. In the end, our 'ontology' isn't a real one and can be assimilated to a simple data structure.

We rapidly get a problem when we tried to store mails and questionnaires. We first thought that we could save to whole structure of the mail and only access to useful content at the moment of the user request for display. However, with this method, we needed to compute the interesting parts of all mails each time we wanted to access it. It appeared that this was a too heavy task and that this was not realistic for a simple display. To solve the problem we decided to store the mail according to two different "facets", which are different representations of the mail. Figure 23 shows a part of the structure we use to store mails. The first facet named "raw" stores the whole mail with its structure (headers, body-parts…). The second facet called "front" is used to store important information, such as mail subject, sender, recipients, sent and received date, mail textual content, images and attachments (their URL). Questionnaires are also represented in the data structure; to do it we have a simple model: we defined a questionnaire as a set of "Questionnaire Atoms". These atoms are composed of a question, a set of possible response and illustrating images.

The functioning of the service is simple, a mail is sent on the mailbox. Our *Mail Harvester* periodically checks the mailbox for new mails. New mails are analysed, parsed and put on our *structure*. The content of mails (text, images, sounds, attachments…) is stored in

specific folders. Once this is done, the service sends back a response to the sources of the mail with the computed link of the “mail” or questionnaire on the site.

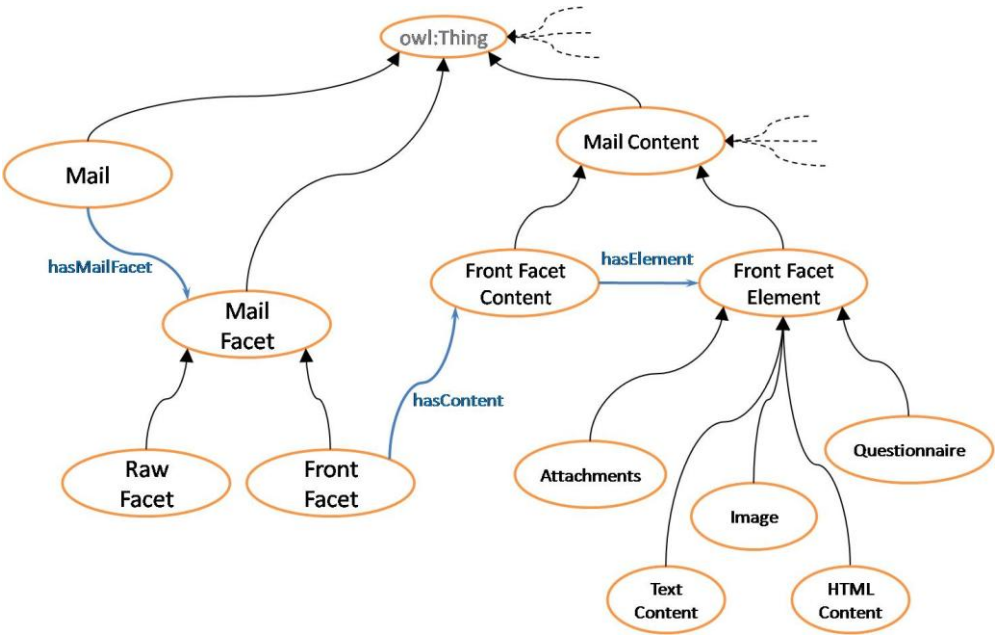

Figure 23 Mail part of the data structure

Once the system has finished its update, mails and questionnaires are directly available on the site. The voting system isn’t complex. A user visiting the website can answer to a questionnaire by choosing a response. The system then updates the stored questionnaire and results are immediately available.

5.3 Evaluation of P-Surveying

We proposed an online evaluation method reflecting the use rate of the system. The idea is that an efficient service tends to be appreciated by users and indeed used. For example in our system it is possible to monitor the questionnaire response rate (ratio between real participant and solicited). We may also compute a utility indicator proportional to the number of participants and proportional to the number of questions. The idea is that it would be difficult to ask directly (e.g. phone call) a lot of questions to a large group. The reusability of previous results may also be monitored. These indicators and others (nomadic ratio...) may be visualised through a graphic board or results contextualised. This will provide a feedback for the services administrator but also for the service users

We also wanted to see how users turn out the services to use it in an unpredicted way, to do it we needed to let users use them for a long time.

To make a first evaluation of our system, keeping in mind the rapid-prototyping method that we used, we published a questionnaire that proposed to ten users. Clearly this evaluation wasn’t meant to provide substantial information for statistical analysis; instead, it was designed to give us interesting feedback on the usability and the main principle of the service. An important point to notice is that doing this way, the evaluated system is also the evaluation system. As it was the first evaluation, we restricted the number of users to ten, but from various origin and social background. The chart on Figure 25 summarizes the results.

The Figure 24 illustrates the flow of the user through the pervasive surveying system. As previously said for our system, the evaluation is made without any external application, consequently the present screenshots were made during the evaluation process.

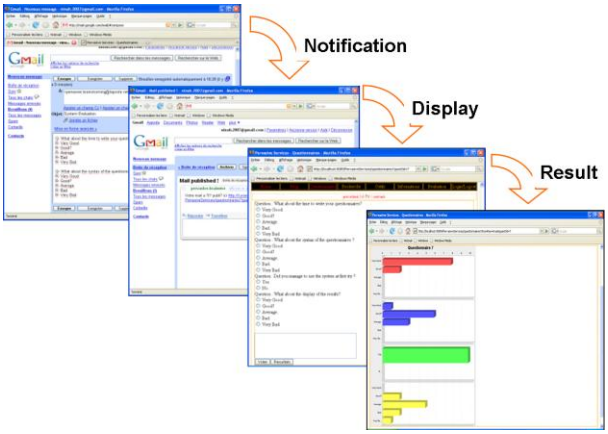

Figure 24 P-Surveying Flow

For this evaluation, we proposed a set of for 4 simple questions:

Q1. What about the time to write your questionnaires?

Q2. What about the syntax of the questionnaires?

Q3. Did you manage to use the system at first try? (yes/no)

Q4. What about the display of the results?

We added another question to ask users which part of the system they think we have to enhance first.

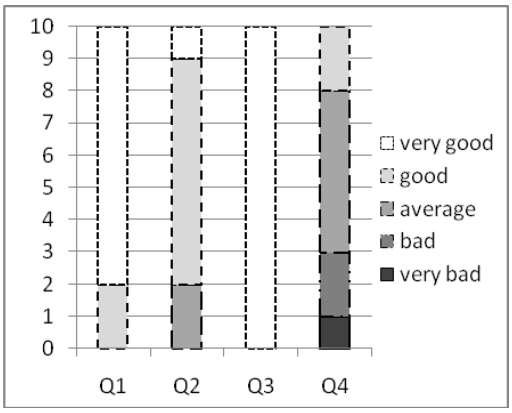

Figure 25 Evaluation Results

This first evaluation point out several wishes and opinion of users:

- According to Q1, Q2 and Q3, the syntax we use seems not too restrictive;
- If we only consider Q3, the system seems simple enough to be used by any kind of users;
- Considering Q4, and as we previously thought, the display is a critical point, and we needed to make efforts to improve it.

To conclude on the evaluation, we can say that the integration of the evaluation as a part of the system itself is a practical method. Combined to the rapid-prototyping principle, they allowed us to make fast, efficient and user-centred evaluation and evolution.

5.4 Synthesis

This first approach, to the computer supported collaborative work and pervasive computing domains, has allowed us to develop an original service offering a simple way for co-workers with different kind of devices to collaborate. Even if this first experience was interesting and more work could have been done to improve the system, we rapidly came to

the conclusion that with the rapidly increasing power and resources of small devices, spending more to time on this kind of system was a loss of time and efforts. Indeed we knew that to properly integrate the pervasive computing and collaborative work we needed to go deeper in the Information Technology levels and focus on a more fundamental research allowing us to consider both pervasive and collaborative aspect in any situations, not in a very specific one. Keeping in mind this perspective, it became clear that we needed to design a model supporting our needs.

Chapter 6 PCSCW Model

Recent progresses in software and hardware technologies have allowed the use of more and more advanced applications and services. It also brought computing capabilities to mobile devices such as smartphones and laptops. This has led to an extensive use of computers to collaborate in some unexpected manners. Among the abundance of models designed to support collaboration some are particularly promising: tasks models, roles models and collaboration's context models. Simultaneously the Pervasive Computing paradigm has emerged from recent researches. In this chapter we propose a model to integrate the pervasive computing perspective into the collaborative work. This integration is proposed by the use of an original model: the PCSCW model (Pervasive Computing Supported Collaborative Work). This model relies on some robust concepts: a role model inspired by some recent works, a classical task model coupled to a precise resource model and the development of device collaboration rules. The resulting model provides a seamless cooperation of devices to simplify, facilitate and channel the collaboration of humans.

6.1. PCSCW Collaboration Model

Keeping in mind the works that have been done in the different domains we're interested in, we propose our own model based on some simple concepts: tasks, actions, roles and resources. The main principle of this model is the following: we rely on the fine description of roles, tasks, actions, resources required and the available devices' resources; then by a simple comparison of required and available resources we can select the right "device collaboration rule" to make devices collaborate seamlessly and facilitate the collaboration of users. We'll now give further details about these main aspects and their use in the process of making devices automatically and smartly collaborate.

6.1.1. Task

The first concept to define is the task. This concept is one of the most popular of the recent researches in collaboration modelling. A task can be defined as a set of actions to be performed in a specified or unspecified order to fulfil the task objective. In addition, a task is not always (and in fact most of the time is not) an atomic one, meaning that it can be composed of several sub-tasks with their own actions and objectives. Moreover, we can point out that the collaboration of people takes place when they need to perform a task they can't or shall not do alone. If this task has to be performed by more than one person, it can be considered has a "shared task" or a "common task".

6.1.2. Actions

Actions can be seen as tasks components. In some perspective they could be considered as atomic tasks, however we think that a task carries its own meaning, actions don't, and that's why we should consider them as sub-atomic components of tasks. To illustrate this idea we can figure that the action "opening a web browser" has no "meaning", but opening a web browser and writing a word in a search engine has its own meaning, it is the task of "searching on the web". The description of actions, in their PCSCW consideration is depicted on Figure 26.

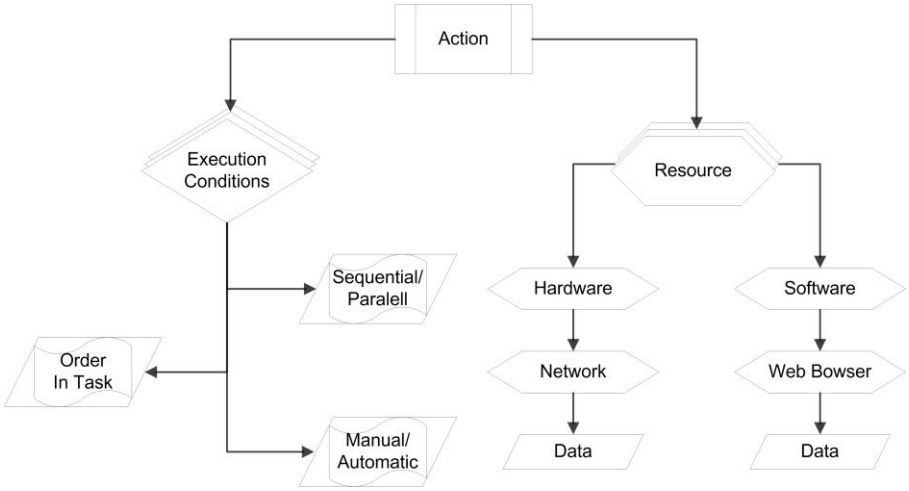

Figure 26 Action Specification

On this figure we can see that actions are defined with the help of mainly two components: execution conditions and resources. While execution conditions contains rather classical information concerning the way actions are handled, resources associated with a given task describe which resources are required for its execution. As we'll see in the following sections, the representation of actions is fundamental in the PCSCW as they hold the representation of resources required to perform them, which by transitivity implies that it allows the representation of resources required to perform tasks containing these actions.

6.1.3. Role

As we have seen previously **[Zhu and Tang, 2007]**, a role can be defined as a set of tasks to be performed by a single entity, giving it responsibilities, rights and duties. A role is not reserved to persons; it can also be played by a group of persons or by an entire organization. Besides, in the same way as a role can be designed for more than one person, a person can play several roles at a time. This is particularly true in the case of a person belonging to multiple groups (for example a work team and a sport club). In addition, a role can have a specific "cardinality" inside of a group, meaning that you can have several people in the same group playing the "same" role. This aspect of the role concept can be confusing if you consider that two people never do the exact same work, that's why roles have not to be confused with peoples.

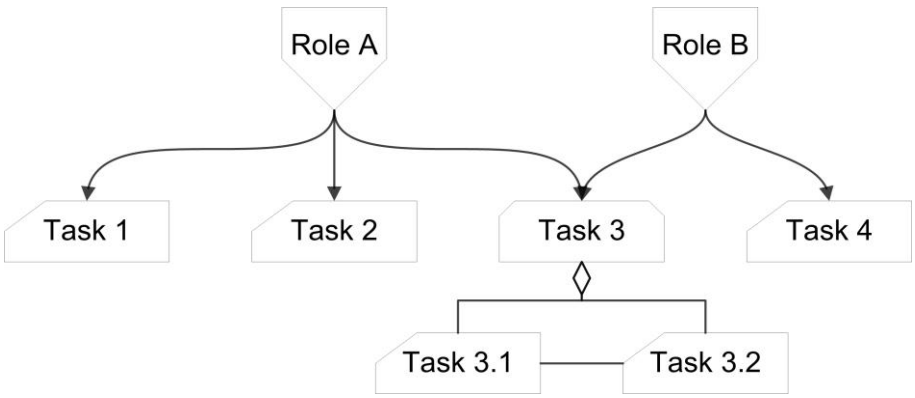

Figure 27 Roles and Tasks

On Figure 27 we give an example of how roles can interact through their allocated tasks. "Role A" has 3 tasks: 2 are dedicated and the third is shared with "Role B". Then this task two is subdivided in 2 subtasks: "Task 3.1" for Role A and "Task 3.1" for Role B. One could have proposed to remove Task 3 and just leave tasks 3.1 and 3.2 affected to their roles.

Still we argue that for some of them it is necessary to preserve links between related tasks. Indeed some tasks require several roles to be completed. For instance the task "writing software specification" is composed of two subtasks: "writing software business specifications" and "writing software technical specifications". Besides, you can't write technical specifications before business ones have been written and they can't be always written by the same person, then the two roles associated to this task will not be held by the same person. In the perspective of context-awareness, the combination of roles and tasks gives more information than tasks alone. Indeed in some circumstances the description of the current task isn't sufficient to have an efficient insight on the collaboration. By having the complete description of the role and its associated tasks, you reach a more complete view of the actual context which can lead, in the perspective of a context adaptation, to a more suited adaptation to this context.

6.1.4. Resources

If you intend to model the context of people in order to develop context awareness mechanisms, at some point of your reflection you will have to face the representation of users' resources relevant for the part of context you're interested in. Obviously in our model, we can't avoid this part, it is in fact one of the most interesting point we want to explore. Indeed, we argue that the description of tasks should be made through the representation of actions and resources required to perform them, giving then the representation of resources required for this task. Going even further we could describe facultative resources that can be effective to perform the task but which are not mandatory. Thus, considering that you've got a fine description of the task (actions and associated resources) a group is performing, you can have accurate indicators of the state of the task. This can lead to a fine monitoring of the task by evaluating the fulfilment of its composing actions in regard of their required resources and the current state of the context. This ability to precisely represent the state of tasks of the collaboration provides the opportunity to stay aware of the collaboration, leading to the ability to develop collaboration-awareness mechanisms.

6.1.5. Smart devices

By extension of the precedent aspect of our model, we propose to associate (smart) devices with tasks. To do it we have to figure out that smart devices are parts of the available resources. Furthermore, it is necessary to have a description of devices capabilities. For instance, if you consider that a high-speed connection to the Internet is required for your task, the best device to support it can be quickly identified by a simple query to available ones or by a more efficient request to some kind of a context manager. Such a mechanism is particularly effective, as it can make several devices cooperate seamlessly. Our consideration of seamlessness in the context of the PCSCW model consists in several aspects: the non-disruption of current devices activity (that is to say: even if the behaviour of devices evolve to adapt to their context, it shall not stop them from continuing their current activity), it also implies the transparency for users (eventual action required to be performed by users should be strictly related to their actual collaboration and not to the cooperation of devices). From this last aspect we can also assume that the collaboration itself has not to be complicated or disrupted, it can only be altered by the cooperation of devices in order to be simplified or more efficiently channelled.

6.1.6. Roles for devices

As a refinement of devices description, we propose to define their roles in the collaboration process and more precisely for a task. Thus, a task becomes the natural link between peoples and devices via the description of roles for both of them. As for "Human Role", the role of a device describes its responsibilities and rights. To take a simple example a device can have a role in a collaboration process giving the responsibility of providing the Internet connection for a given user. With this example we can point out a major difference

between human roles and devices roles, humans' roles are based on actions performing while devices roles are based on resources providing.

The Figure 28 sums it up. On this figure we quickly modelled a simple, but common task: the development of a client-server application which implies the development of a shared object: the communication interface between the client and the server. For this task we need a Collaborative Design Tool and an Instant Messaging Tool, which is not mandatory but can improve the collaboration. An interesting point here is that the model can enhance the collaboration by proposing optional resources such as, in this case, a messaging tool. Furthermore the model itself can be refined by describing precise rules for the messaging tool to be proposed and used; in some cases it can be preferable not to use it. The designing tool is provided by a computer while the messaging one is available on the smartphone. Thus we can say that the computer plays the role of "Heavyweight application provider" while the smartphone has a "Messaging application provider" role, even if these roles don't actually have a name. These are roles *de facto*.

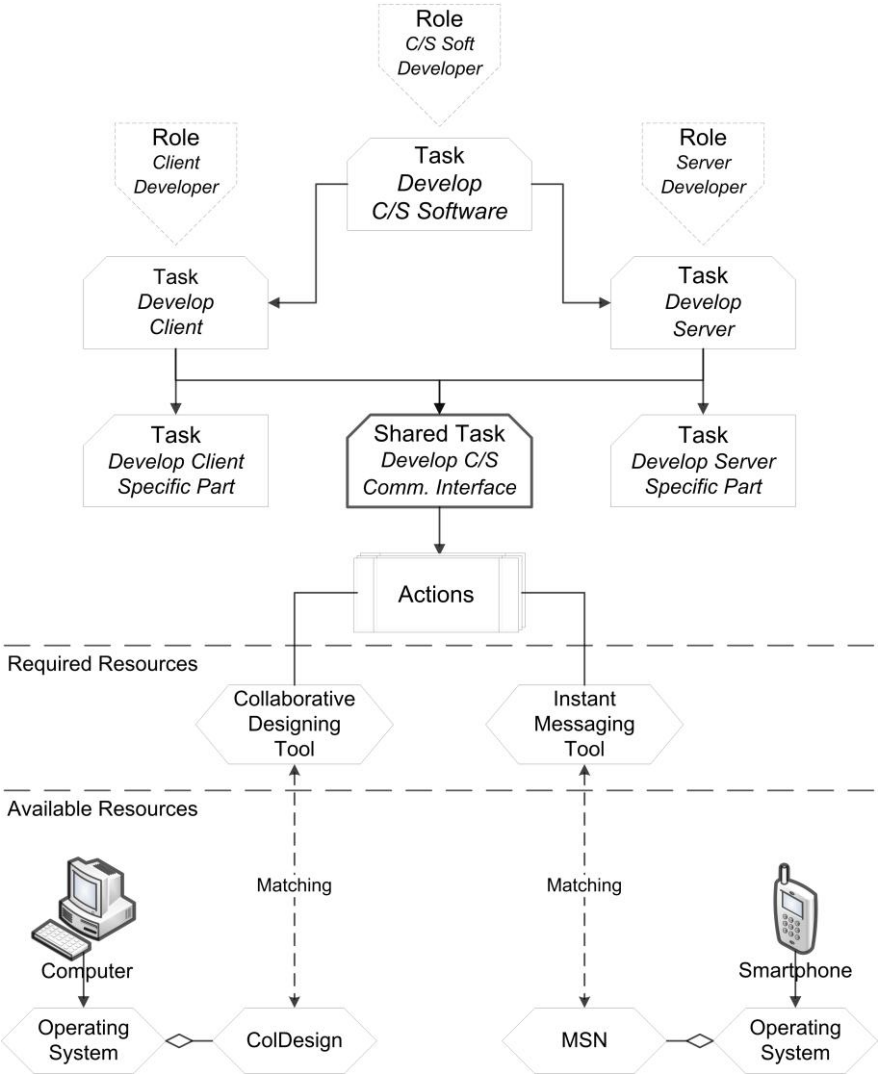

Figure 28 PCSCW model, example of application

All devices don't natively support collaboration with others. In order to solve this kind of issue we argue that the definition of device collaboration rules could be of great help. These rules intend to define actions and tasks that could be automatically performed by devices to collaborate in order to allow a user to complete his own tasks. The main idea behind this is the following: a user needs two (or more) resources to complete an action

related to a task; these resources are not available on a single device, but the combination of several of the surrounding devices can supply the required resources. Thus, device collaboration rules define what actions can be performed by devices to collaborate, in order to finally provide required resources to users. These collaborations can be of various kinds: network access sharing, heavy computing task delegation, and notification of events, anything you can imagine to make several devices cooperate. These Device Collaboration Rules will be illustrated in the following section.

6.2. Device collaboration rules

In this section we will make an in-depth investigation of the devices collaboration rules. The essence of these rules it to allow devices to automatically cooperate to improve the collaboration. This improvement is realized through the facilitation of user's tasks and the eventual automation of most simple ones. To reach this objective, the models we have previously presented serve as a basis on which we can build "device collaboration rules". As we will see in the following sections, rules themselves are based on the description of collaboration (made by the description of tasks, roles, actions, resources and constraints) and the representation of current devices context (which includes collaboration context) in order to in fine to adapt to current context, according to the collaboration needs and possible facilitation of user's work. Let's see how they are designed and how we can use them.

6.2.1 Rules

Making two or more devices collaborate doesn't only rely on resource matching; indeed you need to have defined a set of behaviours to trigger when user's context matches some rules requirements. To be coherent with its main principle, rule behaviour contains actions to be performed with the description of their associated resources. Indeed, each device collaboration rule is defined with the following syntax (1):

$$\text{IF } (context.resources \equiv rule.resources) \text{ THEN DO } rule.behaviour \quad (1)$$

The choice of detecting the potential collaboration of devices with the use of "Device Collaboration Rules" has been made as it offers some advantages in term of simplicity. Indeed, even if some other ways could have been used to detect appropriate behaviours, the representation we propose is probably one of the simplest for creating and maintaining rules, while keeping a decent evaluation speed at runtime.

Besides we need to express a specific need here: all device collaboration rules must have the same knowledge for their reasoning, it implies that context representation has to be "locked" during the reasoning process. This need reflects the fact that if the state of the context (its representation) evolves between the beginning and the ending of the evaluation of rules, it may lead to inconsistent states by the selection of multiple behaviours related to different states of the context.

Obviously, several rules can have a similar or partially equivalent set of required resources, it implies that more than one rule can be matched by the current context and lead to some kind of conflict. Presumably, this kind of situation can happen relatively often and it is critical for a proper functioning of the model to be able to deal with it. The main problem behind these conflicts is the fact that devices will not be able to determine the best behaviour to choose. Then, to be able to select the adequate rule to trigger, we need a tool to evaluate the relative suitability of each behaviour that has been found. To fill this requirement and as we'll see in the next section we propose to define constraints on required rules' resources.

6.2.2 Constraints

The design of Device Collaboration rules for the PCSCW already requires describing roles, tasks, actions and resources. These resources can be of various kinds mainly

categorized along hardware and software ones. In order to be able to complete this design we need to be able to express constraints over the required resources. As we have seen in the previous section, the goal of these constraints is to help decide the most suited behaviour to be used by the device according to its context. A constraint on a required resource has to be able to describe a precise point to evaluate, an expected value for this point and a level of importance for this point. The constraints system we propose is depicted on the following picture.

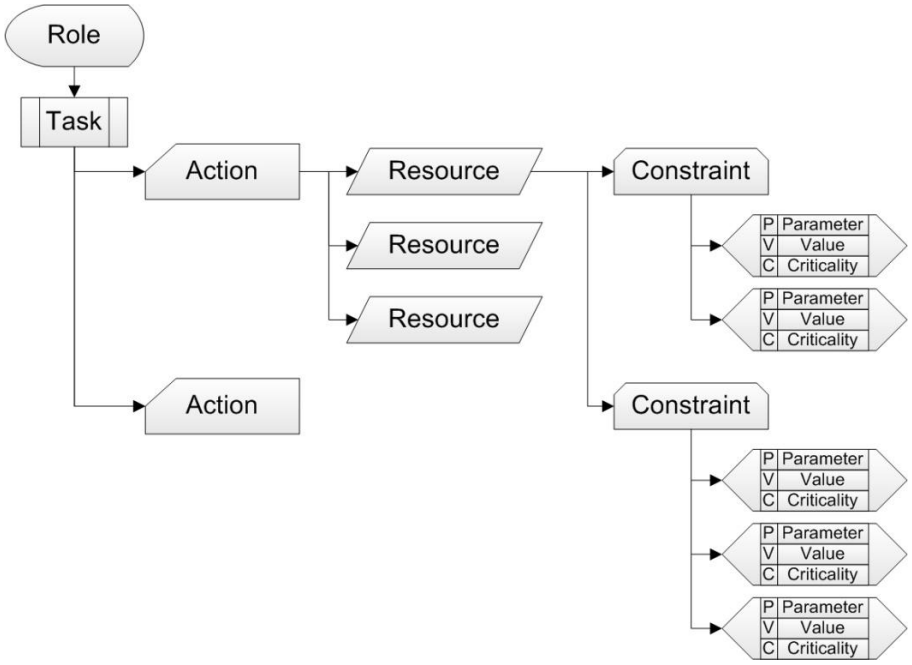

Figure 29 Constraints for the PCSCW Model

Figure 29 defines the addition of constraints on resources of the PCSCW Model. As it is depicted, a single resource may be related to several constraints, each of whom is described by a triplet {P, V, C} as: *P* a parameter which represents the precise point to be evaluated, *V* the expected (or required) value (or threshold) for this parameter and *C* the criticality of this parameter. This last component of a constraint has a specific impact as it is the one allowing a device to select the appropriate collaboration rule.

In addition to this triplet we propose to organize constraints in five main categories, facilitating and guiding rules designer in their work: *Availability, Cost, Privacy, Reliability and Security*. These categories reflect the recurring problems that can appear when dealing with collaborative work and machines interactions. Collaborative work implies availability constraints as the collaboration may require the access to some specific resources (files on a server, shared agenda, etc.). It also implies privacy constraints as the collaboration may take place in business relations or for personal matter (the shared agenda you want to access mays be private and has not to be made available for public access). Collaborative work may also mean constraints on the cost of the collaboration (the access to a shared agenda may not be relevant if the cost is unconscionable). The perspective of machines (devices) interactions also implies different kind of constraints; they mainly differ from collaborative constraints by the simple fact that they are not directly related with the objective of the collaboration rather than with the means of how the collaboration may take place. Availability constraints for machines interactions correspond to resources needed to ensure the proper interaction between them (for instance, a Wi-Fi network adapter has to be present and available for two devices to communicate over this kind of network). Reliability constraints express the need for devices to be able to "trust" specific aspects of their interaction (such as the stability of a server connection, or the quality of context information provided by some sensor, etc.). Finally, security constraints for machines interactions basically correspond to the wish of avoiding potential hacking of theft of data. In this perspective, the security constraints can be

considered both as machines interactions and collaborative work related. Examples of parameters falling below these categories could be: CPU Load (Availability), connection price (Cost), data access (Privacy), website breakdown relative frequency (Reliability) and network secured protocol (Security). Given the fact that values are related to parameters and are illustrated in the following use case, we won't give more examples of them.

While parameters and values are easily collectible from real use case, the criticality needs some more analysis and requires defining its own set of values. In this perspective we propose to use a really simple seven-level scale: *Optional*, *Very Low*, *Low*, *Average*, *High*, *Very High* and *Mandatory*.

Optional indicates the constraints doesn't need to be fulfilled but can provide a valuable benefit for the collaboration and can help choosing between two equivalent rules. On the contrary, the Mandatory level implies that if the constraint is not met the collaboration rule cannot be used in the current context.

6.2.3 Using device collaboration rules

Until now we have described all required concepts to understand the PCSCW model. Let's have a look at the real use of all these descriptive levels and how the model helps at finding the right cooperative behaviour.

To formalize and facilitate the use of collaborations rules we have defined a six-step process describing how a specific rule can be triggered during the collaboration:

1. On context data update, an analysis of this update is automatically started;
2. If this analysis points out that some device collaboration rules may eventually improve the current collaboration by facilitating the accomplishment of an action we start the comparison between context information and rules activation requirements;
3. This comparison can end in three ways:
    a. No rule can effectively improve the collaboration in the current state of the context, we stop the process here;
    b. One rule can improve the collaboration, in this case we jump directly to step 6;
    c. Several rules can improve the collaboration, in this case we need to choose between them the most relevant and efficient, we go to step 4;
4. To choose between the selected rules we need to compare their suitability, their relative efficiency. To do it we confront action's required resources and their associated constraints with resources supplied and used in each rule' behaviour.
5. From this confrontation we bring out a score for each rule and we just need to keep the rule with the higher score. In the case where several rules have the same score it means that none of them can be "automatically" preferred and the device has to take one of them arbitrary.
6. Run the chosen behaviour.

Hence to compare two selected rules we need to quantify each of them according to resources and constraints. In fact, the way we have defined constraints facilitates this comparison by relying on the quantification of criticality and the evaluation of the constraint fulfilment of each rule.

Resource	*Res_1*		*Res_2*	…	*Res_N*			*Suitability*
Constraint	*$C_{1,1}$*	*$C_{1,2}$*	*$C_{2,1}$*	…	*$C_{N,1}$*	…	*$C_{N,M}$*	
Rule R_A	$V_{A,1,1}$	$V_{A,1,2}$	$V_{A,2,1}$	…	$V_{A,N,1}$	…	$V_{A,N,M}$	$S_A = \sum V_{A,I,J}$
Rule R_B	$V_{B,1,1}$	$V_{B,1,2}$	$V_{B,2,1}$	…	$V_{B,N,1}$	…	$V_{B,N,M}$	$S_B = \sum V_{B,I,J}$
…								…
Rule R_X	$V_{X,1,1}$	$V_{X,1,2}$	$V_{X,2,1}$	…	$V_{X,N,1}$	…	$V_{X,N,M}$	$S_x = \sum V_{X,I,J}$

Figure 30 Rule Comparison Matrix

Figure 30 depicts the rule comparison process. On this matrix, each rule to be compared is evaluated along with its provided resources and their constraints. Thus, for each constraint of each resource we assign a value ($V_{r,i,j}$) which is limited by the criticality of the constraint evaluated ($V_{r,i,j} \leq Crit(C_{i,j})$). As for now we have decided to use a simple system: a value is comprised between 0 and 5, a *very low* criticality means 1, *low* means 2 and so on until *very high* which means 5. As we already evoked, an unsatisfied mandatory constraint eliminate the rule, while an optional one can only be used to decide between two equivalently suitable rules. Then if we consider a constraint with a *high* criticality the evaluation of this constraint can't be higher than 4. We know that the assignment of values can be sometimes problematic if the threshold value that was first defined in the constraint is not easily comparable. Thus, if we consider constraints such as *data encryption* it can be hard to give a value to a different encryption method than the one that was defined. Still, there are solutions to this kind problem, for instance we can use predefined rankings.

Limiting rules' assigned values with the associated criticality and relying on constraints *Values* prevent from selecting a rule that does not satisfy most of the most critical constraints but tremendously outperform a minor one. Thus, even if network bandwidth constraint has been defined with a *low* criticality and a value which has to be at least 0,5mbps, the rule allowing a ultra-high speed connection faster than 100mpbs but poorly satisfying other constraints will certainly not be selected (except for the case where other rules are worse) as its connection can give it more than 2 "points".

An obvious drawback of this method is the potential non-trigger of useful rules. Indeed, we can assume that in some specific contexts several rules can be needed to efficiently fulfil some tasks. However, the response to this problem comes from the model and the rules themselves: once the device has chosen and applied a specific rule, it may rerun the reasoning engine to find other rules to apply. With this simple mechanism a device can chain its context adaptation and apply all the rules needed.

Finally we obtain the suitability level of each rule by adding up all assigned values. This level corresponds to the level of adequacy between the resources involved by the different behaviours and the constraints defined of resources of the collaboration. Once this suitability level has been computed for each rule, the last step to choose between them is simply to compare those levels and find the highest one. In the end, the addition of the constraints system on the rest of the PCSCW model allows to automatically select the most suited behaviour by computing a suitability level for each rule matching the current state of the context.

6.2.4 Use Case

The description of the model and device collaboration rules' process is now complete. In order to illustrate their use and benefits we propose a use case based on a simple collaboration between 3 co-workers.

Leela, Amy and Philip are members of a team and have to collaborate on a new marketing campaign for the new product of their company. In this perspective they have to perform several tasks together. Let's suppose that they have to make a brainstorming session to design a new advertising board. Amy is working at their main office, but Leela and Philip are not physically present. Leela is working at her home while Philip is in mission in Kenya.

In order to be able to work at the same time Amy has sent invitations to Leela and Philip for a virtual Brainstorming with a dedicated software at 3 PM (GMT). In a "device consideration" Amy is working on her usual workstation, Leela has its personal laptop, Philip on his side has a tablet-pc and a smartphone. At 3 Amy has started the server part of the application and has connected her station. At the same time Leela's laptop and Philip's tablet-pc need to connect to the Internet in order to be able to join the Brainstorming platform. To do it they rely on the PCSCW model that should allow their devices to make the right decision.

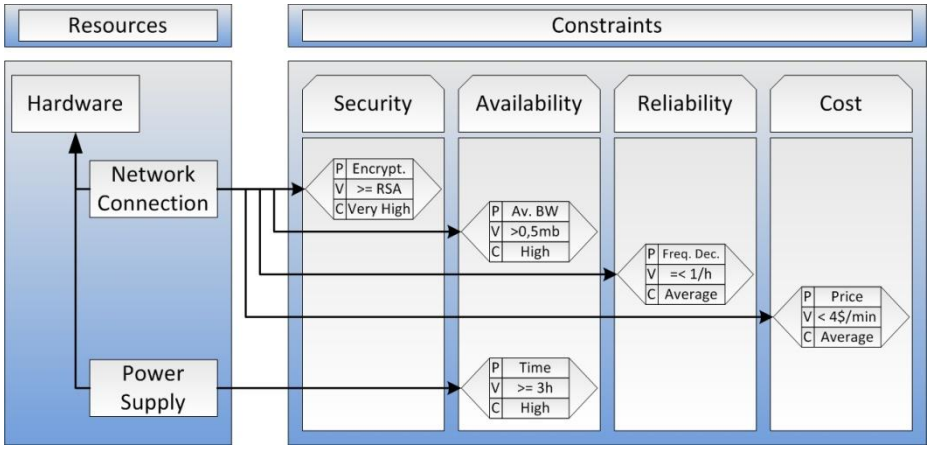

Figure 31 Internet Connection Constraints

The task associated with the brainstorming activity described with the PCSCW implies several constraints on the resources used by the devices. Figure 31 defines and describes the set of constraints associated with the "Connect to Internet" action. On the left side we've got the set of resources required to perform the action and on the right their constraints. For this specific action there are two resources: a network connection and a power supply. For the network connection we've got:

- 1 security constraint: the encryption has to be at least RSA[14] this constraint has a Very High criticality as the collaboration taking place is close to confidential;
- 1 availability constraint: the average provided bandwidth has to be at least 0,5mbps, this constraint has a High criticality as the application can work with less bandwidth but user's satisfaction and experience may be dramatically lowered by such limitation;
- 1 reliability constraint: the probability to experience network disconnections has to be less than 1 per hour. As this point doesn't completely stop the collaboration it has an Average criticality;
- 1 cost constraint: the price of the connection has to be less than four dollar a minute. As it doesn't obstruct the collaboration this constraint has an Average criticality.

As for the network connection we also have a constraint on the power supply resource:

- 1 availability constraint: the energy supplied has to be sufficient to maintain the connection for three hours in order to have enough time for the brainstorming session. This constraint has a High criticality.

Leela's laptop hasn't many choices and connected itself to the Wi-Fi access point of Leela's home's ADSL modem.

As depicted by Figure 32 Philip's situation is totally different. In addition to the tablet-pc, the smartphone and the hotel Wi-Fi access point, we've got a description of resources required for the connection to Internet. It also depicts the three possible scenarios to establish the Internet connection:

- Direct connection of the tablet-pc through the satellite network adapter;
- Connection of the tablet with hotel's access point;
- Connection of the tablet with Philip's smartphone with a connection bridge between cell-phone's Wi-Fi and 3G networks to allow the tablet to access to Internet.

Each one of these possibilities has advantages and drawbacks:

[14] RSA patent: http://www.google.com/patents?q=4405829

- Direct connection with satellite network:
  - Advantages: highly secured, only rely on tablet's energy, relatively stable;
  - Drawbacks: slow connection (~0,2mbs) and costly, occasional disconnections;
- Connection to hotel's Wi-Fi:
  - Advantages: good bandwidth(~2mbps), free, low energy consuming;
  - Drawbacks: poorly secured (WPA), variable bandwidth, disconnections every fifteen minutes;
- Connection with smartphone:
  - Advantages: as we use ad-hoc Wi-Fi the security is up to the two devices and can be relatively good, the average bandwidth is fair (~1 mbps) and the connection is relatively stable;
  - Drawbacks: power supply is limited by smartphone's battery life which is limited to 2.8 hours due to the high energy consumption of the 3G and Wi-Fi adapters.

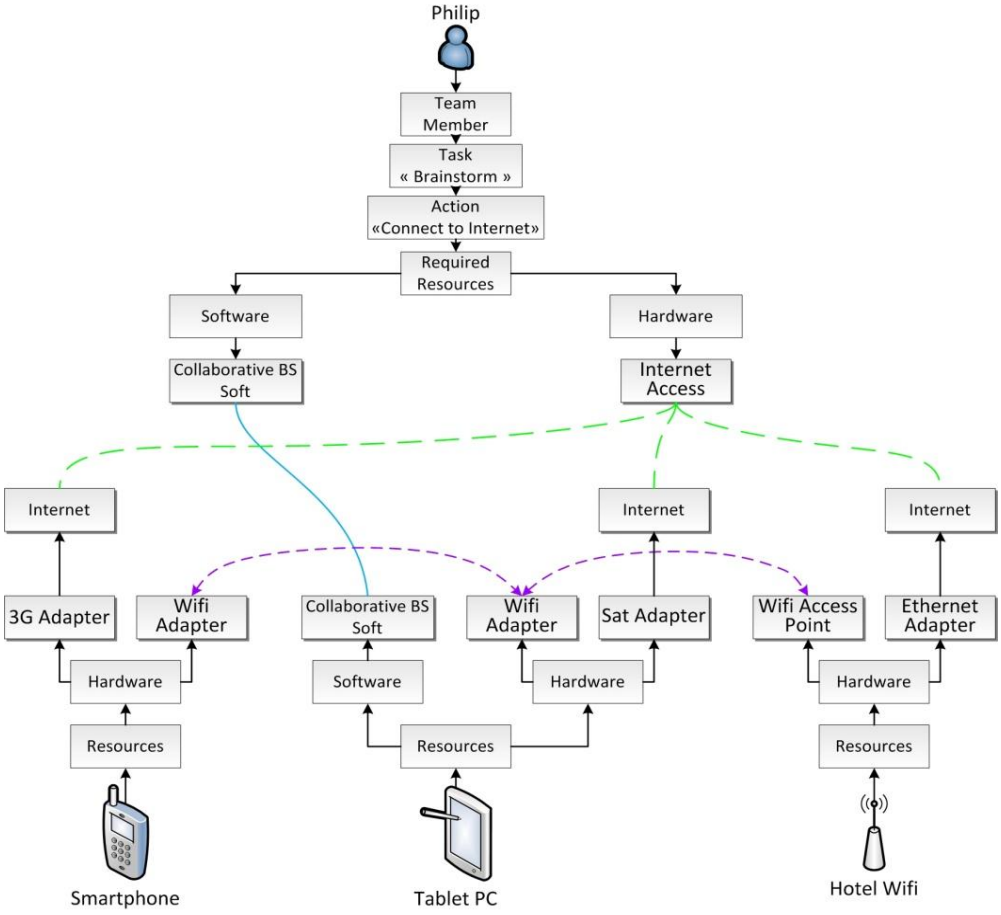

Figure 32 Philip's Digital Environment

We consider that Philip's tablet has already acquired all these information; he must now find the best solution. This is simply done by analysing solutions constraints fulfilments such as displayed on Tab 9.

	Network Connection				*Power Supply*	*"Score"*
	Security	*Availability*	*Reliability*	*Cost*	*Availability*	
Satellite	5	2	2	0	4	13
Hotel Wi-Fi	1	4	0	3	4	12
Smartphone	4	4	3	2	2	15

Tab 9 Comparison of Connections

In our case, even if the scores are relatively close, the smartphone scenario offers more advantages than others as it combine a good security, availability and reliability for a limited cost. For the battery life, as it is not a mandatory constraint given that the smartphone can maintain the connection during more than 90% of the desired time with the eventual possibility for Philip to simply plug its charger.

Finally, after having evaluated the situation, Philip's tablet decides to establish a Wi-Fi ad-hoc connection with his smartphone and create a bridge between the 3G and Wi-Fi connections. Philip can now connect to the brainstorming platform and collaborate with Amy and Leela.

6.2.5 Pervasive Surveying through PCSCW

In this last part of the presentation of our model we'd like to describe how our first work around the pervasive computing and CSCW could have been represented with the help of our model and how it could have been supported. Let's remember the situation:

- The manager of a team wants to have the opinion of its team members about a specific topic (for example about a project he's planning);
- As his team is often spread over different locations, he can't meet each of them physically;
- To solve issues they can encounter for this collaboration, we proposed a service based on the use of mails and forum to channel the opinion of the team and provide efficient synthesis of the group opinion;
- The system itself is based on the automatic publication of multiple-choice questionnaires which can be sent by mail to a dedicated mailbox, mails are then analysed and contained questionnaires are published on a forum where team members can vote and give their opinion.

If we consider this use case with our model we can distinguish two roles: the manager role and the basic team member role. The manager role has a cardinality of 1 while the team member role has an unspecified cardinality for this group. The team itself is mapped to a "Group Role" with its own set of tasks. The manager role allows its player to perform a "Survey" task while team members are allowed to perform the task "Answer Survey".

Let's consider the case where a member of the team, Bob, is equipped with a laptop and a cell-phone, both switched on. His laptop has only a Wi-Fi and an Ethernet adapter without available network in range. On the contrary, his cell-phone is connected to a HSDPA network and has its own Wi-Fi adapter (but as for the laptop, without access point available). Bob's manager has just sent a new questionnaire; an automatic mail is sent to him with a link to the published questionnaire. In the traditional case, Bob should open the mail, go on the forum and access to the questionnaire on his phone. We suggest that this interaction could be dramatically improved. Let's consider that Bob is deeply focused on his laptop and that his cell-phone lies at some distance of him. Here a simple but still efficient device collaboration rule can take place:

- When receiving a new mail on cell-phone;
- If User is working on superior ergonomics device (Computer, Laptop …) which can be connected with cell-phone and which hasn't Internet access;
- Then perform tasks: establish a connection from cell-phone to computer, create a bridge between cell-phone connections and notify user of the new mail.

Figure 33 represents what resources are necessary to perform the "Bridge Connection" task. As we can see on the following figure our model serves at representing resources of users' task to find matching devices but also to determine if those devices can cooperate to supply the desired resources. Indeed to perform the connection between two devices we have defined the required resources: an active connection on the device that have

to bridge it; and a network adapter of the same type on both of devices. In our case the available connection can be found on the cell-phone as the HSPDA one while common network is supported by Wi-Fi adapters. Once the cell-phone and laptop are connected the bridge can be activated.

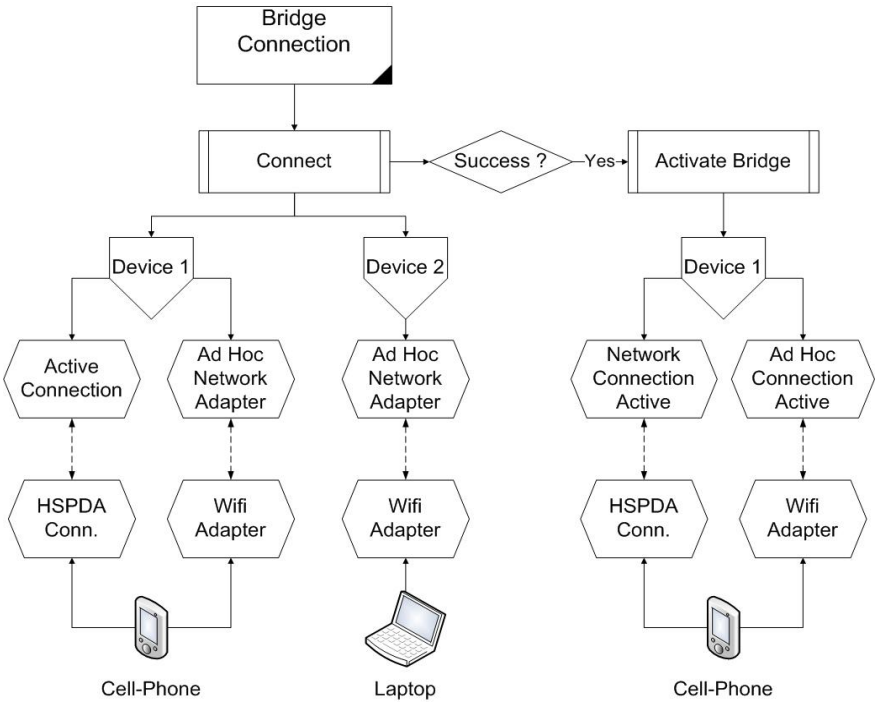

Figure 33 Device collaboration, connection bridging

These scenarios show how our model enables the efficient cooperation of surrounding devices to enhance the collaboration of users. Besides, this type of scenario can be further extended to multiple users and devices, creating a real “pervasive and collaborative network”.

6.2.5 Synthesis

The model we propose does not come out of nowhere, it relies on robust researches that inspired us and guided us to develop it. As it has been intensively mentioned, our model is based on the notion of roles, for people and for devices. In a moral consideration, the representation of roles is not a substitution of the representation of a person, it is only a part of a person, otherwise one can quickly come to the conclusion that only roles matters and peoples don’t. But from a model perspective taking into account the role as a variable can help to apprehend the complexity of a pervasive environment. In such context, roles or resources can vary depending on spatial, temporal or collaborative constraints. Having a model with which could provide precise information on the collaboration, supporting a more precise evaluation of the "efficiency" of the collaboration, could also be a useful tool for designing purposes.

The PCSCW model is designed to facilitate the collaboration of users by making devices cooperate. In its nature this model could be considered as a meta-model as it tells how some sub models can be used and combined to improve the collaboration of users. Even though we can take out some benefits and drawbacks of our model put in regard with previous ones.

The most obvious benefit of our model compared with others is the fact that it natively considers the distribution of resources and the possibility to use them all at once. Indeed most of traditional collaboration models, based on tasks, roles or even more advanced collaboration awareness models focus on the way to keep users and their devices aware of the collaboration. We think the PCSCW model is going further in this direction by using

collaboration awareness to enable the “collaboration intelligence” of devices and develop their proactive behaviours. Another noticeable and valuable benefit of our approach is the possibility to precisely monitor the current state of the collaboration. Indeed, as we have to depict each task and their related actions it helps channelling the collaboration awareness.

Indeed, our approach can seem very descriptive, detailed and requiring great efforts to be used. But we want to take an advantage from this issue. In fact, all awareness mechanisms do not require the same level of description. For some of them, only the top levels are relevant. This is why we argue our model is able to describe and reason on different granularity levels, from a simple description of devices until a fine description of each object manipulated by an application on a virtualized operating system. Thus, we can say that our model naturally supports the scalability of awareness mechanisms by its adaptability to the description of resources. This scalability can even bring an abstraction capacity by allowing designers to represent high-level information and reason on it. Besides, this scalability advantage is twofold, it allows the description of resources with various granularity, but it also offers the possibility to reason with few context information and then when computing resources are limited or information are hard to obtain.

This work relies on two main aspects: the representation of required and available resources and the description of device collaboration rules. Still we know our work has its own disadvantages. One of the mains is the need to create these rules. Indeed to adapt to a specific context it requires having a more or less generic set of rules. Even if this particular point can seem annoying it can be a source of improvement. Despite rules have to be written before the use of the model, they can also be derived from user’s activity, preferences and constraints dynamically.

Chapter 7 Evaluation of Pervasive CSCW

"*New evaluation strategies are needed that uncover central issues associated with groupware success and failure, and they need to be more flexible than they currently are in order to adapt to a greater range of factors that need to be considered*" **[Neale et al, 2007]**.

In this chapter, we want to emphasize the necessity of a comprehensive evaluation taxonomy and strategy for applications in CSCW. Dealing with the evaluation of any computer-based system requires different types of knowledge: you have to know the domain related to the system, you have to know how the system works technically, but you most certainly need to know how to evaluate such system. Even if some systems can be particularly hard to evaluate due to their domain and technical complexity, the hardest part of the evaluation often consists in the determination of how you will evaluate them. As we have seen in the dedicated section, there are numerous ways to evaluate a system, even more associated methods but still no effective way to help you know what method you should use.

In this perspective we proposed a taxonomy of evaluation methods helping us to characterize them. Exploring further this idea, we propose to organize evaluation of CSCW systems in Evaluation Strategies that provide the evaluation process for a given system, defining what kind of method to use at what time. With this solution we intend to be able to plan almost any evaluation of CSCW system. In addition to the strategy building methodology we propose guidelines to find indicators for the evaluation of pervasive collaborative systems.

7.1. Taxonomy

The first element we consider to build our taxonomy is the fact that it can't be composed of a simple dimension. On the contrary, it should be designed according to a complex space. However, all the dimensions of this space are not mutually orthogonal, implying that some of them can partially overlap themselves. The second step of this process is the identification of important aspects of evaluation methods. It gives us the list of characteristics that will be used in the taxonomy. The third step consists in defining the "meta" aspects of CSCW evaluation; this step is done by analysing the previous list of characteristics and extracting the main categories. Fourth, we sort the characteristics according to these categories with the possibility to have a given characteristics in several categories (but obviously not all the characteristics in all the categories). Fifth, inside the categories we try to gather characteristics by discovering similarities between them and then building sub-categories. These five steps have led us to the following taxonomy (Tab 10):

✓ *Development specific aspects:* - *Development process:* ➢ *Type:* • *Iterative;* • *Extreme Programming;* • *…* ➢ *Development Step;* ➢ *Goal:* • *Maintenance;* • *New System;* • *…* - *Final System:* ➢ *Goal;*	✓ *Method specific aspects:* - *Goal:* ➢ *Focus:* • *Usability;* • *Quality;* • *Performance;* • *Sustainability;* • *Utility;* • *Coherence;* • *Extensibility;* • *Scalability;* • *…* - *Type:*

- *Scalability;*
- *End-users type*
  - *Anyone;*
  - *Developers;*
  - *Government;*
  - *...*

- *Current System:*
  - *Scalability;*
  - *Step in development process;*
  - *Automation capacity*
  - *Evaluation Cost:*
    - *Computational Cost;*
    - *Human Cost;*
  - *Feature to evaluate:*
    - *Feature type;*
    - *Feature maturity;*
  - *Evaluation needs:*
    - *Modus operandi:*
      - *Exploration;*
      - *Evaluate some precise points;*
    - *Evaluators type:*
      - *Experts;*
      - *End-users;*
      - *Developers;*
      - *Representative Sample.*

✓ *Collaboration specific aspects:*

- *Collaboration Model:*
  - *Mode:*
    - *Asynchronous;*
    - *Synchronous;*
    - *Mixed*
  - *Group Structure:*
    - *Size;*
    - *Scalability;*
    - *Members Coupling;*
    - *Members type:*
      - *Scientists;*
      - *Developers;*
      - *...*
  - *Evaluation Cost:*
  - *Formality:*
    - *Formal;*
    - *Informal;*
  - *Business consideration:*
    - *None;*
    - *Weak;*
    - *Average;*
    - *High;*
    - *Full;*
  - *Users multiplicity:*
    - *Single;*
    - *Multiple;*
- *Cost:*
  - *Time Cost;*
  - *Human Cost;*
  - *Computational Cost;*
- *Evaluation context aspects:*
  - *Evaluators type:*
    - *Experts;*
    - *End-users;*
    - *Developers;*
    - *Diversified;*
    - *Representative Sample.*
  - *Evaluation Place:*
    - *Laboratory;*
    - *Real location;*
  - *Evaluation Step:*
    - *Preliminary;*
    - *Main;*

• *Human Cost;* • *Time Cost;* - *Evaluation focus:* ➢ *Single user behaviour;* ➢ *Multiple user behaviour;* ➢ *Mixed.*	

Tab 10 CSCW Evaluation Taxonomy

The proposed steps for building the taxonomy are not mandatory; their main goal is to provide guidance for us and users in the representation of evaluation context and is largely inspired by classical taxonomy and ontology construction methods such as Bachimont in **[Bachimont, 2000]**.

Obviously this taxonomy isn't exhaustive; it does not intend to address every methodology with all their details in this presented form. However, it can be simply *extended* to support any new kind of method.

7.2. Strategy

As we mentioned previously **[Herskovic et al, 2007]** propose an interesting approach to CSCW evaluation by proposing a three-phased strategy. This strategy relies on the principle that you don't need the same method at each step of your development:

1. *Formative lab-based methods* (perform some pre-evaluation to avoid main errors).
2. *Field methods* (with the participation of users associated context).
3. *Qualitative methods in real work settings* (evaluate if it really works).

This work is one of the few we found to propose a real strategy of evaluation. It is even more valuable as it builds a frame for evaluation, meaning that instead of telling which method to use, is only give a more general category. Then you're free to use the best method for your system, picked-up in the right category.

Relying on this good idea, we decided to go deeper in the definition of strategies refining the description of evaluation methods according to the development strategy, processes and steps. Naturally this refinement takes also advantage of our previously presented taxonomy.

Another point has to be noticed before diving into the strategies. We think that the good evaluation of a CSCW system has to be organized in three phases:

1. Evaluate the collaborative aspect;
2. Evaluate the business aspect;
3. Evaluate the combination of collaborative and business aspects.

This separation is particularly relevant as it allows identifying quickly and efficiently lacks in collaborative and business aspects of the system. Thus is can also help finding problems emerging when you integrate collaboration into the business domain.

Choosing a specific method instead of another is a critical need. It determines if the evaluation you'll lead is relevant or not for your system. Deciding and choosing between these methods is puzzling. Methods have their own weaknesses, and trade-offs, they can be complementary or exclusive. Because the methods found overlapping problems, we expect that they can be used in tandem benefiting from each other, e.g., applying the discount methods prior to a field study, with the expectation that the system deployed in the more expansive field study has a better chance of doing well because some pertinent usability problems will have already been addressed.

To be quite exhaustive our methodology offers two possibilities: picking up an existing strategy related to an evaluation close to your own; or building your own strategy "from scratch". The first opportunity is only interesting in some rare cases where you really are in the hurry and every hour or even every minute count. For the moment we focus on the second one.

So, how can we efficiently choose a strategy to evaluate a given system? Our underlying idea is to find the best matching between the definition of your current system and the definition of the context in which strategies take place. For instance, the strategy will not be the same if you are at the beginning or at the end of your development lifecycle.

This process is done according to five steps:

1. Describe the context of the evaluation.
2. Define the different phases of your development.
3. Extract the Evaluation Strategy Outline from the development phases.
4. Refine Evaluation Strategy Outline's methods' description.
5. Select Strategy's methods.

The first step to build a strategy is describing the context of the evaluation. In order to do so efficiently and exhaustively we propose to take a top-down approach. By this we intend to start by defining high level categories of the taxonomy and then going deeper and deeper. For instance, one of your first elements to describe is the development process type: do you use a traditional iterative process, a waterfall one or do you prefer the Extreme Programming methodology. Obviously, describing the development process is not sufficient, you also need to specify other system's relative aspects, business specific and collaboration's ones. Indeed, as the evaluation process is split in three different phases: business, collaboration and business + collaboration, we need to describe them sufficiently to find appropriate methods for each of them. Moreover, we believe in the proposition made in **[Herskovic et al, 2009]** to separate the evaluation in three phases: short lab experiment to detect main problems, field method with users' context to evaluate deeper and real settings evaluation to gain qualitative feedbacks.

Once the first description phase is done the second step consists in the description of the different phases of your development. For this step you have to define the different steps needed in your process and define the order in which they appear. For instance you should define the order in which you develop the different features of your system and of what types they are, business specific, collaboration specific or else. This part of the specification is really important as it is the base of the evaluation's outline generation.

Third step is the extraction of the evaluation's outline. As we've just said, this step relies on the description of development phases. Thus, to build the outline, or skeleton, of the evaluation strategy we have to consider each phase of the development process and establish if it requires an evaluation phase, moreover, we have not only to consider the development phase alone but we need to consider it and its position in the whole process of development in order to refine the evaluation methods and correctly establish if additional evaluation phases are needed. For instance, if the precedent phase of development was to create storage module and the current one focus on the development of an event logging feature, we not only have to test the new feature, but also the interaction between the storage and logging parts; for example to see if events are correctly represented in the repository.

The fourth step to build the evaluation strategy consists in refining the evaluation methods types selected to form strategy's skeleton. This part of the building process relies on the skeleton and on the previously description made through the taxonomy. Hence, the previous step gave us a set of evaluation methods types described with some broad criteria related to the development process phases. To complete this we use the description of the evaluation context made in the first step. Thus, we only have to "complete" the description of

each method with the evaluation context and then access to a refined description of the evaluation context of each evaluation phase of the skeleton, giving us a refined skeleton.

The last step to build the strategy is the final selection of adequate methods. The main principle of strategy building is to find a matching between the evaluation context and the context in which a method takes place. As the evaluation context is described, we just have to find the corresponding methods. The natural way to perform it is by describing such methods. Hence the "matching" we propose rely on the "comparison" of system's context against methods' intended situation. Thus the description of evaluation methods through the taxonomy is crucial. But it's also a heavy task requiring a long study of each method. Still, that's not an unfeasible work as we think the definition of methods can be enriched by all users, closing the loop of collaboration.

Besides, the taxonomy approach of our work enables users to only describe some general aspects of methods, resulting in a broader range of selected methods of system evaluation but saving large resources in exchange. By doing so, you load your burden with an extra task: choosing between a set of methods.

Finally, you've got an evaluation process consisting in an ordered sequence of evaluation methods: an Evaluation Strategy. Figure 34 sums up how you can build your evaluation strategy.

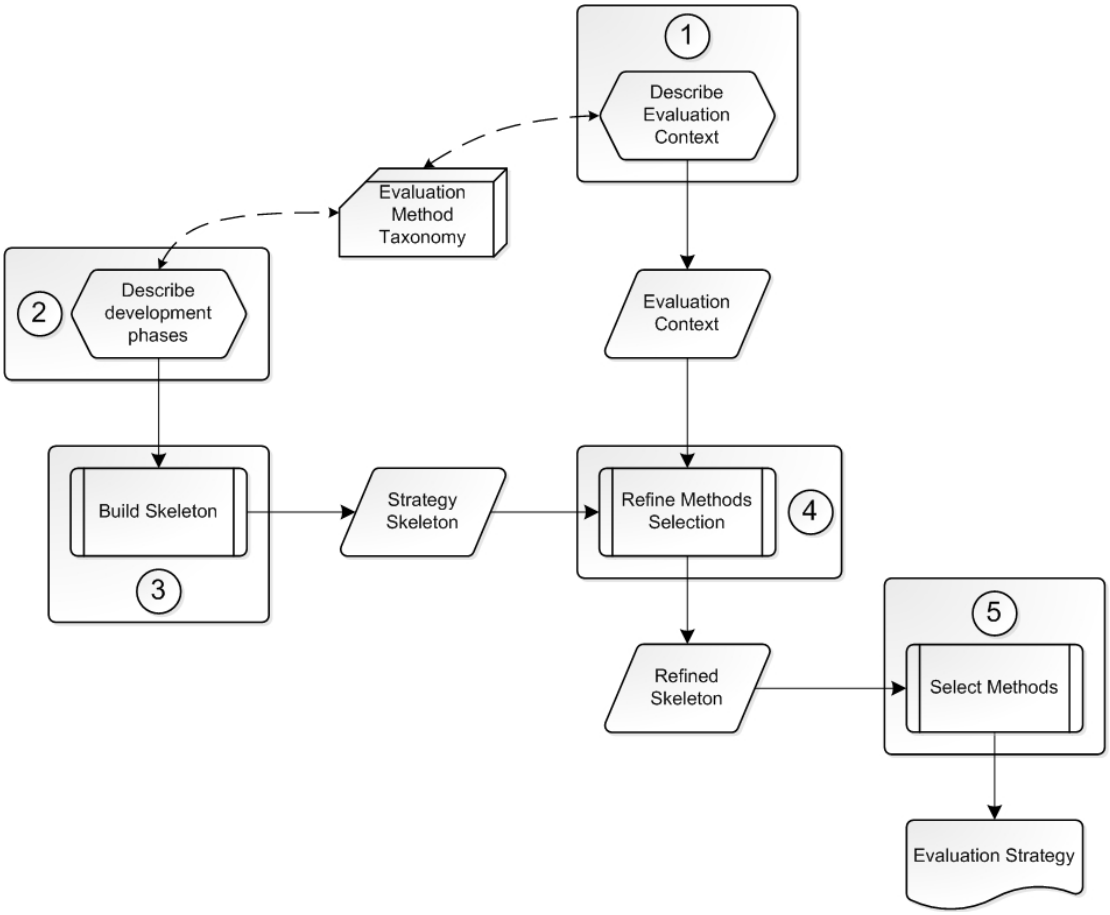

Figure 34 How to build an Evaluation Strategy

To conclude this section we'd like to consider a critical point in the evaluation process which is often a source of conflicts: the lacks of adaptability of evaluation method to the evolution of development process. That is to say, if your development process suddenly speeds up, the heavy evaluation strategy you have chosen may not be able to make it. Thus, we think it is essential for a strategy to be able to be adapted to the changing context. In this perspective, the taxonomy we propose is central. Indeed, it provides a simple tool to find what

methods have to be removed and which have to be used instead in your strategy to fit the new evaluation context.

7.3. Example of Evaluation strategy

To illustrate the use of our approach, we'll now take an example of CSCW system evaluation. Let's consider a service we developed, presented in chapter 5. This service is quite simple: it provides the capacity to automatically publish a questionnaire on a dedicated forum-like website. Thus it allows a team of users to efficiently communicate and collaborate by giving them to possibility to send questionnaires, answer to them and have a synthetic view of the responses even if they only have a low-resources device.

✓ Business aspects:	✓ Collaboration
- Questionnaire ➢ Editing ➢ Sending ➢ Publishing ➢ Viewing ➢ Commenting ➢ Voting ➢ Synthesising	- Questionnaire ➢ Voting ➢ Commenting - Messaging ➢ Notification of publication ➢ Sending - Role management

Tab 11 Evaluation use case - Business and collaboration aspects

As we'll obviously not write the full specification of the system we'll only focus on main aspects of this development. Following the five steps we have defined, the first one is the description of evaluation context. In a first time we have to identify business specific aspects and collaboration ones as shown on Tab 11. To complete the description of the context in which this development takes place, let's make a short description of the resources:

• Human	• Time	• Hardware
○ 20 Man-day ○ 4 Peoples	○ 2 weeks (firm)	○ All necessary

Tab 12 Evaluation use case - Resources

On the previous table (Tab 12) we can see that the evaluation process have to be completed within two weeks. The hardware is not really a problem as required resources are relatively limited. Finally, the team has freed the equivalent of 20man-day to lead the evaluation to be distributed among 4 peoples.

Sticking with the taxonomy we can make the following assumptions:

- The development process relies on a fast iterative method;
- The goal of this process is to create a new feature for an existing system;
- End-users are accustomed to use communication means and web browsers;
- The evaluation have to explore the system in addition to validate the new features and check if it doesn't interfere with the normal behaviour of the system;
- Evaluators of the system need to be end-users for the final part of the evaluation;

Considering the limited time granted for evaluation but the rather large amount of human and hardware resources, the proposed methods have to be adaptable for larger groups with intensive evaluation instead of small groups or loose evaluation sessions. Moreover, to be loyal with Herskovic's proposal (**[Herskovic et al, 2007]**) each evaluation step is subdivided into three phases, but in the case of iterative development process and even more

when the development has to be fast, each loop on a same feature reduces the evaluation time and especially the time for lab experiment.

Secondly, we had to define the development lifecycle we use. Figure 35 shows how this development was organized: Analysis of supporting system in order to know if it was able to correctly support the new features, specification of the new features, multiple development phases, finalization and then delivery.

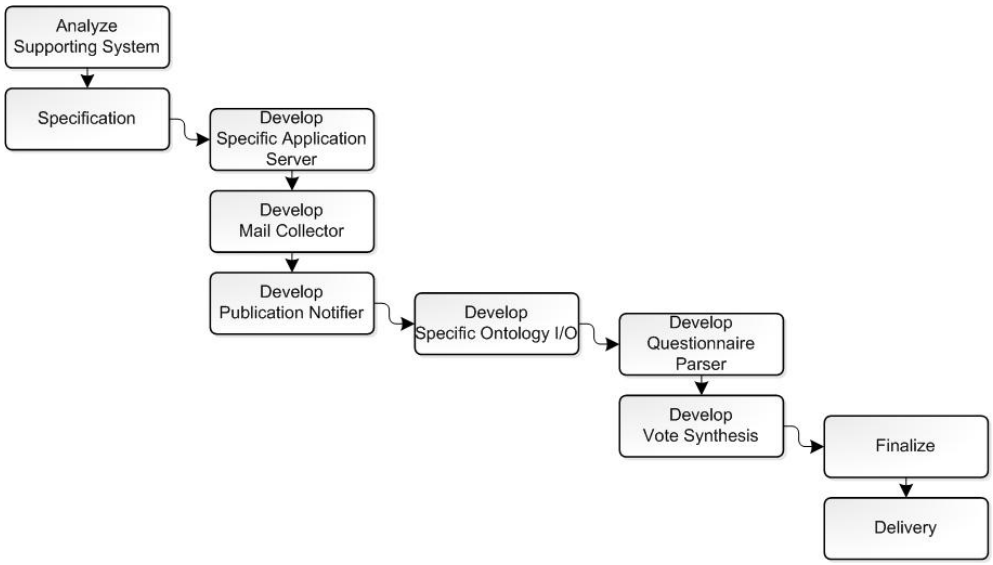

Figure 35 Development Life Cycle

Based on this development life cycle we can extract the outline of the evaluation strategy. As we can see on Figure 36, to extract this skeleton we start by considering each phase that has been described earlier, for each one of them we figure out if it requires one or several evaluation stages. For instance in our example the "Analyse Supporting System" phase requires to evaluate if the system is suitable for the desired evolutions. As a direct consequence we deduce that we need three "Integration Feasibility Methods", one to evaluate if the system can handle the collaborative aspect, one to know if it correctly sustains the business part of the new features and finally a method to evaluate how the combination of these two aspects interacts with the existing system.

Fourth step of strategy building we have to refine the description made in the previous step with the help of the evaluation context defined in the first step. As it would be a little too long to describe all the methods evoked in Figure 36, we'll only refine one of the stipulated method. In order to have a relevant example, we consider the last evaluation step: "Full System Evaluation Method".

Looking at the associated description we are able to see that this step has the following requirements: *Qualitative evaluation*, *Triple-phased* (laboratory, field study and Real Conditions), *Exploratory* (as all features has been tested previously we only have to figure out the overall quality of the system in its wholeness and if some unexpected challenges are leveraged*)* and obviously the evaluation concerns the *Full System*. Considering the evaluation context we know the selected method have to be rather fast than exhaustive as the evaluation time is short. Besides, we need to make end-users participate to the evaluation in order to have a real qualitative feedback but also to be able to efficiently explore the system with users' habits. Considering these requirements we can refine the method to the following ones: 1- Users' Exploration (let users use the new features without guidelines, just with the instructions to know how to use it and let them explore the system); 2- Scenario Based Evaluation (write scenarios to guide users in their walkthrough); and 3- Scenarios Refining Method (starting from some pre-written scenarios, users have to collaborate with peoples in charge or the development to refine scenarios through their own experience and desire). As

we design it, to complete this step of strategy building you have to refine all the methods of your strategy skeleton.

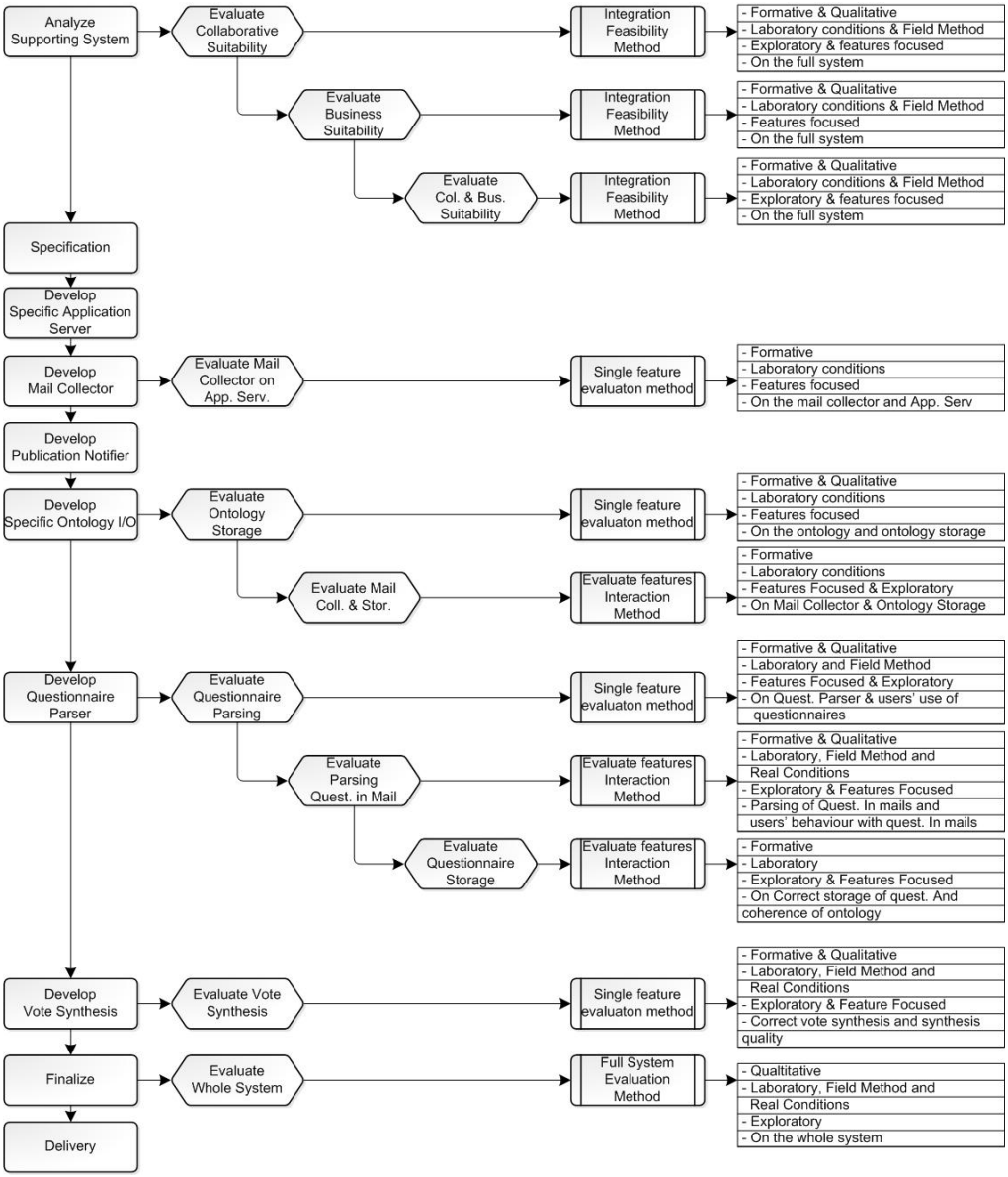

Figure 36 Use case - Strategy Outline

The last step of this process is the final selection of methods. It has to be said that this step is not mandatory for all the evaluation steps of the skeleton. Indeed, for some of these steps you can have found only one method matching their evaluation context. In that case you obviously don't have to make a choice. Nevertheless, given the refinement level of an evaluation step (have it to be desired or not), it can be matched by several methods. Finally, with all the required selections done, we've got our Evaluation Strategy.

7.4 Evaluation Indicators

Among the methods we have studied a number of them are able to perform the evaluation of collaborative work without worrying if it takes place in a pervasive environment or in a more "classic" environment. However we found no method for evaluating specific aspects of pervasiveness within the collaboration. Indeed in our knowledge there are no indicators, measurement or method for automatically tell whether cooperation between different devices of users' environment has allowed or not to improve their collaboration.

Considering this lack we decided to investigate the evaluation of computer supported collaborative work taking place in a pervasive environment. More precisely we focused on the search and definition of indicators allowing the evaluation of such systems. These indicators shall allow evaluators to quantify different aspects of a collaborative system which are: simplicity, effectiveness, reliability, security and quality. They also constitute a base supporting the comparison between two different systems and different settings of the same system.

7.4.1 Indicators

As we focus on the evaluation based on indicators, we try to answer the question: what types of indicators are likely to reflect the collaborative work in a pervasive environment?

Figure 37 provides a general overview of the various indicators that we propose, their "context" of use and their purpose. The centre of this schema shows the collaboration in pervasive environment as it is represented with the model PCSCW. As we said previously, this model provided a basis to represent the roles, tasks, actions and resources of a pervasive work environment. In addition to these concepts we have the representation of users and machines taking part in various collaborative processes.

The overall functioning of the model is as follows: by comparing the resources required to perform a task with available resources of various machines in the environment, it follows a strategy of cooperation between these machines to facilitate the work of the user and so to help their collaboration. From this representation and the analysis of the different needs, we propose a set of indicators and measures that reflect the collaborative work taking place in an intelligent environment.

The first differentiation to make is to separate indicators allowing a “unitary” evaluation which means they don’t require the completion of one task and indicators allowing a consolidated evaluation of collaboration. The consolidated indicators imply an additional constraint in the evaluation: the fact of not being able to evaluate some aspects of collaboration in "real time". Indeed a number of them will be significant only after the end of the collaboration.

To help the definition of indicators we need to describe the various aspects to be evaluated. For this step we have defined a set of five aspects which are measured by quantitative indicators:

- *Simplicity*: This aspect summarizes the fact that the collaboration of users must be easily decomposed into atomic elements;
- *Efficiency*: Along similar lines, yet complementary, efficiency must be at the core of the evaluation. Even if a system allows easy collaboration for users, it should not prevent the purpose. Efficiency can be measured by the rate of tasks completion without errors, the execution time task compared to the medium time recorded by a satisfaction index, etc.
- *Reliability*: Another very important point in any system and maybe more in a collaborative work system where various users are required to work together, reliability ensures that the system will work consistently. This criterion can be measured by the difference (e.g. measured by number of steps) between the objective reached and the medium objective found (i.e. usual goal);
- *Security*: Security is a vast topic area, in the case of security collaboration it can be expressed in various ways: access to resources, confidential data encryption, anonymity, etc. The problem of security is increasingly at the centre of attention, it is even more critical with Pervasive Computing. Indeed, by the mere fact of allowing machines to communicate so that the user does not need to know, it leads to new security problems. The different devices of environment interconnect more

intensively than in a "classic" environment, they are therefore exposed to greater risks;

- *Quality*: The last aspect to evaluate is the quality which can hardly be equated with the union of all the previous aspects; however we think that some other additional indicators are needed to evaluate the quality itself. Quality can also be reflected by a weighted sum of each criterion above.

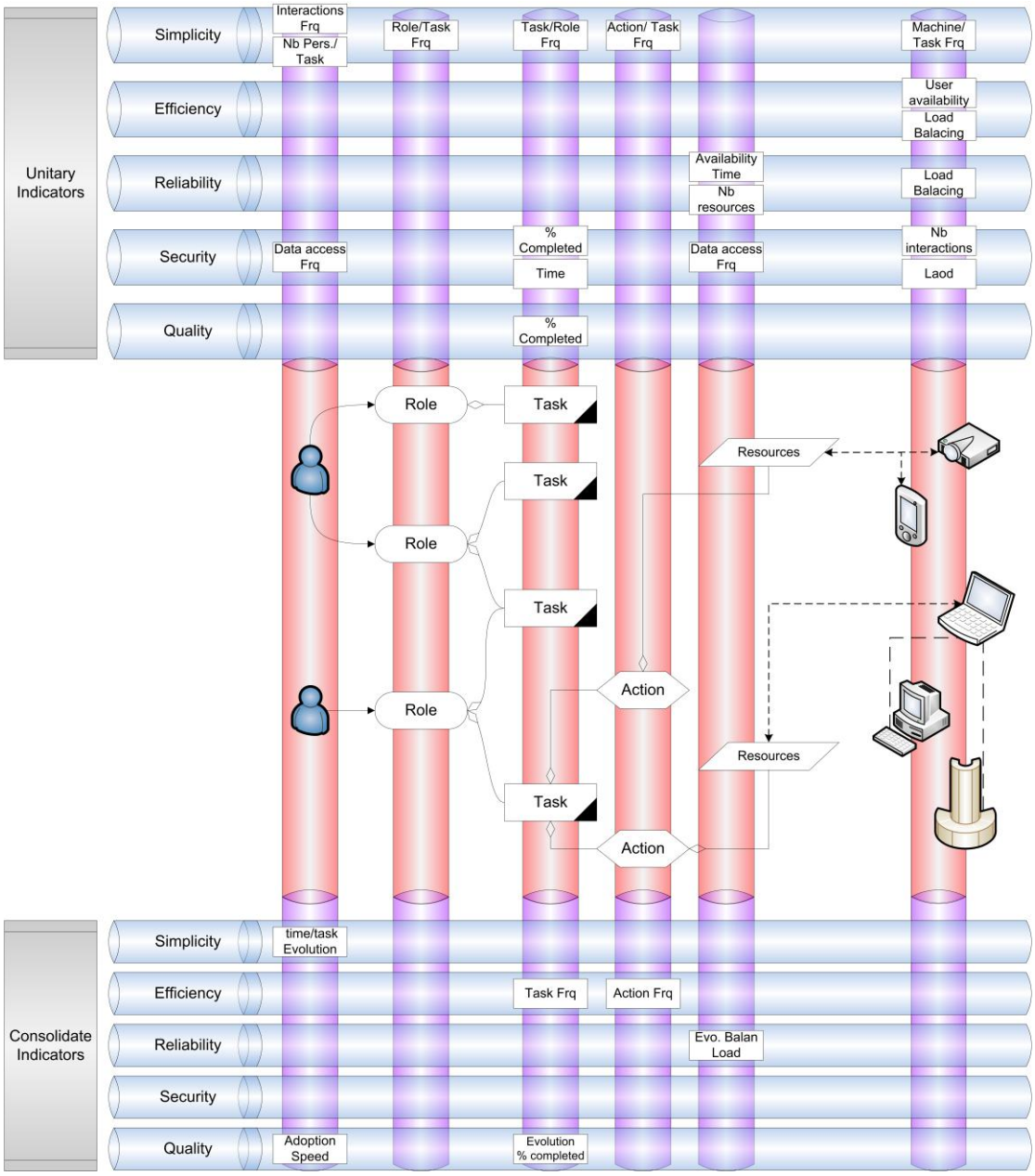

Figure 37 Evaluation Indicators and their context

Once the main aspects of evaluation have been defined, we can give more details about the indicators we propose. Let's start with unitary indicators.

- **Simplicity** :
  - ***Users*** :
    - *Number of people taking part in the collaboration*: the more people there is taking part, the more the collaboration and its coordination are complex;

        - *Number of interactions between humans*: similarly to the number of humans taking part, the number of interactions between them is a good revelatory of the collaboration complexity;
        - *Number of interactions between humans and machines*: indicates the number of interactions between humans and machines that have been necessary to complete a scenario or a given task. Naturally, the lower this indicator is implies a greater simplicity for users;
    - ***Roles*** :
        - *Number of roles by tasks:* this measure is somehow similar to the number of persons taking part to the collaboration, except for the fact that it indicates the number of different roles instead of the number of persons, as a person can play multiple roles and a role can be played by several persons. The measure of this indicator can be significantly different from the number of person;
    - ***Tasks*** :
        - *Number of tasks by roles:* contrary to the precedent measure, this one indicates the average number of tasks a role has to play. This specific indicator can be useful to evaluate the intrinsic complexity of a given role;
    - ***Actions*** :
        - *Number of actions by tasks*: this indicator evaluates the relative complexity of tasks of the collaboration. It can be specifically useful when you have to find ways to fasten or improve collaborative tasks as you may want to find most complex tasks;
    - ***Devices :***
        - *Number of devices by task/by action*: measures how many devices have taken part to the fulfilment of a task or an action;
        - *Number of human-devices interface*: enumerates how many different interfaces a human had to use during the collaboration. This really specific indicator is particularly interesting as the complexity of numerous collaborative systems comes for a great part from the lack of homogeneity of applications and from the proliferation of various interfaces it involves;
- **Efficiency** :
    - ***Roles*** :
        - *Percentage of roles having fulfilled all their tasks* : simple measure of the number of roles which have been able to end all their tasks during the collaboration;
    - ***Tasks*** :
        - *Global percentage of accomplished tasks*: this indicator allows quantifying the proportion of completed tasks. This indicator can be measured over a limited period of time or not;
        - *Task accomplishment time* : indicates the relative effectiveness of the system by telling how much time has been required to complete a specific task;
    - ***Machines*** :
        - *User availability*: this measure corresponds to the ratio of time where a machine is available for the user. It can notably help knowing if the pervasive aspect of the system doesn't obstruct user's actions. For example if a person has to use an heavy application on a device while this same device is already performing others processes for another user, he can be disturbed in his actions;
        - *Load Balancing*: in the same perspective, this indicator allows us to know if processes are correctly dispatched over the different available devices. By

combining this indicator with the precedent we can accurately evaluate if loads are correctly balanced on the "right" devices. For example, it is logical to give heavy processings to a high-performance server, while giving them to a smartphone isn't.

- **Reliability** :
  - ***Resources*** :
    - *Resources Availability*: by measuring the time of availability of resources requires by action of the collaborative work, it is possible to estimate the level of reliability of the system. Indeed, the lower the availability will be, the higher the risk of collaboration failure will be;
  - ***Machines*** :
    - *Load Balancing*: in addition to its use to measure the effectiveness of the collaboration, it can also help deducing the reliability of the system. Indeed, the less the load balancing is effective, the more the risk of collaboration failure is elevated.
- **Security** :
  - ***Users*** :
    - *Amount of data accessible/required:* this indicator reveals the number of data accessible to the user and in which measure these data are necessary to complete his tasks. The larger the access is, the higher the risk of accessing restricted ones is.
  - ***Roles*** :
    - *Roles/User adequacy:* this indicator shows in which measure the different roles of the collaboration have been played by the adequate users. Thus, the fact of playing a role of another person can induce severe security problems;
  - ***Resources*** :
    - *Amount of data accessible/required:* as well as for users, we have to avoid exposing critical and/or confidential data;
  - ***Machines*** :
    - *Number of devices interactions*: the security implies to protect sensitive data and thus to limit the possible accesses. In this perspective it is critical to limit the number of interactions between devices. Indeed, the more devices are communicating, the more they expose their data to others;
    - *Load*: an excessive load can compromise the general security of the collaboration by allowing unauthorized accesses due to applicative errors;
- **Quality** :
  - ***Tasks*** :
    - *Ratio of completed tasks*: beyond the simple efficacy, this indicator gives us information about the general quality of the collaboration.

Let's consider now the consolidate indicators; contrary to the unitary ones, they need a deeper analysis and can require advanced statistics in order to be significant.

- **Simplicity**:
  - ***Users*** :
    - *Evolution of task execution time* : this indicator allows us to compare the required time for users to complete a task according to the different collaborations situations;

- **Efficiency** :
  - ***Task/Action*** :
    - *Task Frequency*: this indicator corresponds to the apparition frequency of a given task during the collaboration. The more a task appears, the higher is the necessity to automate it (if possible);
    - *Action Frequency*: same as precedent, but for action.
- **Reliability** :
  - ***Resources*** :
    - *Evolution of load balancing*: in some pervasive systems, as learning is a major aspect of these systems, the functioning of devices can evolve. Considering this, we can observe disturbing evolutions where some devices can become overloaded;
- **Quality** :
  - ***Users*** :
    - *Adoption speed*: indicator common to numerous domains, the adoption speed of a collaborative tool is critical. Indeed, if a single user can quickly and simply change its use and then change of tool, it is different for a collaborative system where it is necessary to find a solution satisfying most of users;
  - ***Tasks*** :
    - *Evolution of task fulfilment:* unlike the corresponding unitary indicator, the evolution of task fulfilment ratio across the different phases and collaborative experiences allows us to have a good perception of system's quality.

7.4.2 Indicators Aggregation

In the previous section we defined a set of indicators to quantify the evaluation of various aspects of collaborative work. However, the use of these indicators alone would be sufficient to fully evaluate a system for collaborative work. Even if the unitary indicators provide the raw data on an "instance" of collaboration while consolidated indicators provide a quantification of the changing aspects of the collaborative work, it is necessary to go further to identify potential problems and measuring the efficiency of solutions. In this context, it seems necessary to introduce an additional concept which is the aggregation of indicators. In practice, this aggregation occurs in comparing and contrasting different indicators. This should enable us to make statistics to highlight correlations between data, to ultimately lead to quantification and a more accurate evaluation system for collaborative work in pervasive environment.

We can take various examples of aggregation. First we can simply combine two indicators such as number of roles by tasks and number of tasks completed, thus having a more accurate evaluation of the efficiency of the division of tasks within the collaborative environment. We can also compare the time required for completion of tasks with the number of machines taking part in the collaboration. Another example of aggregation of indicators could be a comparison between the evolution of time needed to accomplish a task and evolution in the rate of machine availability.

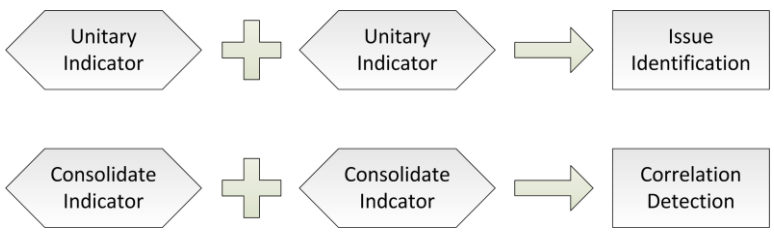

Figure 38 Indicators Aggregation

In Figure 38 we summarize what we have formalized as an aggregation of indicators. We can see that by combining two unitary indicators it is possible to identify potential problems. On the other hand, by combining two consolidate indicators we obtain a simple way to detect correlations between different aspects of evaluation.

Overall the aggregation of the proposed indicators can expand the coverage of evaluation and give a more accurate and more reliable quality of collaborative work.

7.5. Synthesis

The evaluation of collaborative systems in pervasive environments is a relatively new aspect of computer sciences. Actually they're hardly few works intending to consider this problem. Given our implication in the integration of the pervasive computing within the computer supported collaborative work, it was a natural step for us to consider their evaluation. In this perspective our contribution is twofold.

In a first time we have proposed a methodology allowing systems developers to plan and organize the evaluation of their system according to their development process and the context in which the evaluation has to take place. In an effort of reusing previously developed evaluation methods, our methodology relies on the use of a taxonomy helping the representation of these methods according to their characteristics. The outcome of this methodology is the production of an Evaluation Strategy respecting the evaluation constraints and suited for the development process.

Our second contribution for the evaluation of pervasive and collaborative systems is the definition of indicators helping evaluators quantifying the real impact of pervasive aspects over the collaboration of users.

Chapter 8. Simulator

The validation of a model is a highly critical phase. It requires efficient tools to determine if the model effectively works in the way it was meant to. For our work we have to validate a model allowing multiple devices to cooperate in a human collaboration environment. Given this complexity it was clearly impossible for us to validate the model by conducting a live campaign. Such campaign would have required too much time and resources. Thus we decided that the most efficient way to validate our model was to develop a simulation tool providing us relevant information about the new behaviour induced by the use of our model. In this chapter we will detail the design and the development of our Simulator. In a first place we will present the needs analysis and the specifications that were used to design this simulator. After that we will give a general view of the underlying architecture before going deeper into the design. Finally we'll describe the use of the simulator.

8.1 Needs Analysis and Specifications

In this first part we focus on the requirements of the simulator, what features have to be present in order to facilitate the validation of the model. This part is really important as it describes how the simulator should work, and what we wanted to do with it.

8.1.1 Simulator goals

The main goal of this simulator is to allow us to simulate the behaviour of devices during the collaboration of users. This simulation has obviously to take into account the PCSCW model and its use by devices to adapt their behaviours. In order to represent the simulation itself we rely on the use of scenarios. These scenarios have to represent every details of the simulation from the initial situation to its end by the completion of at least one collaborative task. Thus each scenario contains, in addition to its initial state, a set of events that will be triggered as the simulation evolves. Each of these events can then trigger new behaviours of agents (humans or machines).

This first goal only tells us that this tool has to be able to simulate a scenario and interactions between humans and devices. But this simulator doesn't only have to support the unfolding of the scenario, it also has to give us information about the unfolding. Indeed, we need to be able to get information about the simulation, how many interactions have occurred, how much "time" it took for the simulation to unfold, what were the precise durations of each collaborative task. Thus, it has to be able to provide us indicators of the simulation and of the collaboration.

8.1.2 Simulation Features

From the goals we have presented we can figure out features the simulator has to support. Obviously the most basic features are the representations of persons, devices and the general environment. This quite simple assertion is not so simple to apprehend. Indeed, the simulation of the environment implies that the system has to be able to deduce logical implications of devices behaviours. For the simulation of agents (devices and persons) it requires that their representation in the system has to be independent and that they can behave simultaneously.

According to the second main goal of the simulator, it has to be able to produce a report on the unfolding of the scenario by giving us the relevant feedbacks on the simulation. This report has to be easily treatable after its production. This would allow us to retrace the evolution of the simulations according to the evolution of settings and the modifications of behaviours.

8.1.3 Additional Specifications

As we wanted to have a user-friendly interface we wished to have a minimalistic graphical user interface, allowing us to, at least, load and run scenarios from a given repository.

8.2 General Architecture

We have defined our needs and the main specifications of the simulator, then, and before diving in the details of the simulator we have to present the general architecture we designed and the technological choices we made.

8.2.1 The PAC Architectural Pattern

In order to facilitate the development of this simulator we rely on the use of the PAC[15] design pattern. The use of this pattern can also guide us in the development by providing a fixed way to organize components. PAC stands for "Presentation, Abstraction, Control".

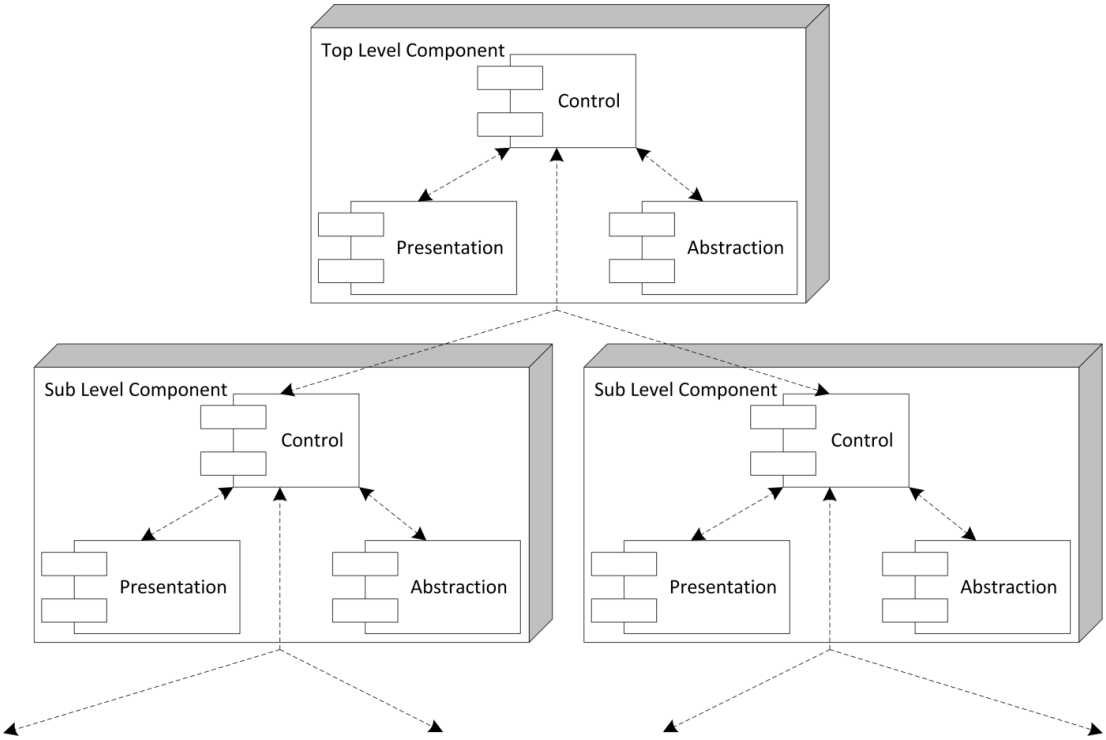

Figure 39 The PAC Architectural Pattern

This architectural pattern is in some ways similar to the MVC2[16] pattern. As we can see on Figure 39 the main principles of the PAC pattern are the following:

- Components are subdivided according to the classical presentation/control/data (abstraction) perspectives;
- Presentations and Abstractions are not aware of each other as they cannot communicate directly;
- Presentations and Abstractions can only communicate with their associated Controller;

[15] PAC: http://en.wikipedia.org/wiki/Presentation-abstraction-control

[16] MVC2: http://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93Controller

- Components are organized hierarchically and only communicate through their respective controllers;
- Components of a same "level" cannot communicate directly and have to interact with the controller of the above level.

Given that we had decided to use the PAC pattern we needed to make a specific implementation of it. In this perspective we defined the base shown on Figure 40.

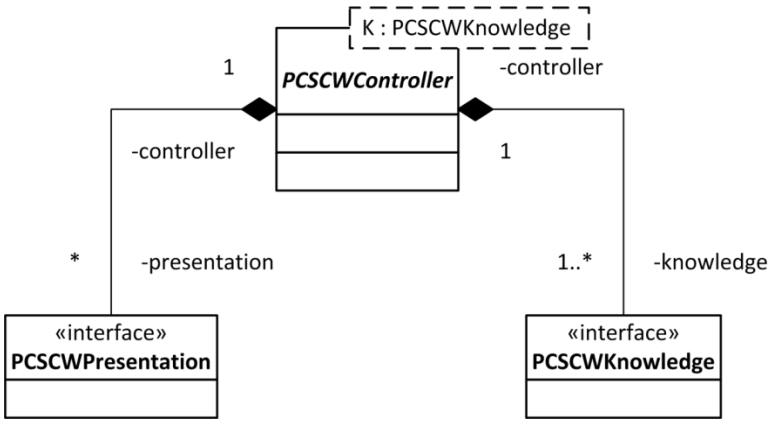

Figure 40 Base of the PAC Implementation in the Simulator

As we can see we defined two interfaces "PCSCWPresentation" and "PCSCWKnowledge". The "PCSCWController" abstract class is defined as a generic class with a parameter K. This parameter has to implement the PCSCWKnowledge interface. This class is generic as we wanted subclasses to specify the type of PCSCWKnowledge used to manage their data.

Indeed, the module managing the environment of the simulation and the module managing an agent will not have the same kind of knowledge and then won't access to them in the same way.

8.2.2 System overview

Now that we have defined the architectural pattern we'll use to implement each module of the simulator we can consider the organisation of those modules. On Figure 41 we describe how the main modules of the system are organized and how they are related.

Thus we've got the following four main modules:

- *Simulator*: this main module is the first to be launched. It has to manage the basic interface of the simulator where the user can load the scenario he wants from a given repository. Then it also has in charge to create the scenario engine that will later manage the simulation. This module only communicate with the scenario engine;
- *Scenario Engine*: as previously said it has in charge to manage the simulation. It loads a scenario from a given repository and initiates the environment and agents of the simulation according the setting of the scenario. It also manages the unfolding of the scenario. This unfolding is done by the firing of scenario events. The scenario engine can communicate with the simulator, the environment and agents of the simulation;
- *Environment*: this module is used to manage the environment of the simulation. It has to maintain the representation of all knowledge of the context of the simulation. Another primordial role of this module is to trigger environmental events according to the behaviour of agents and the unfolding of the scenario. The environment can interact with the scenario and agents

- *Agents*: obviously the "final" goal of the simulator is to simulate the behaviour of agents (persons, devices). Thus these modules will manage agents of the simulation, their knowledge, their behaviour, their interactions. They can communicate with each other and the environment.

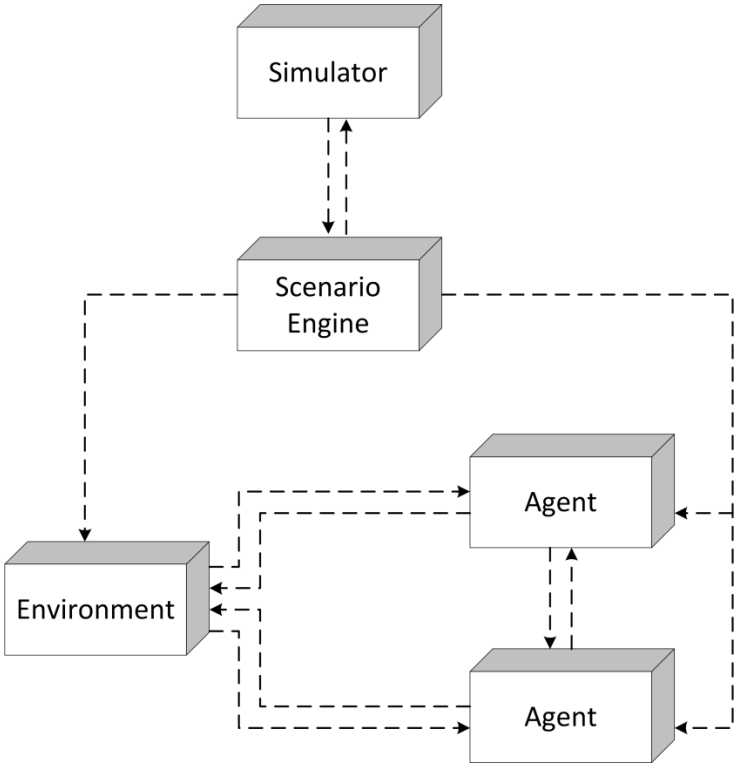

Figure 41 System's modules

As we have mentioned, the environment and the agents cannot interact with the simulator or the scenario engine directly. However, as we'll see later, the environment and the agents can send events that can be intercepted by the scenario engine and the simulator.

8.3 Technical choices

In this subsection we will see what technical choices we have made in order to design and develop the simulator. The development of such tools is relatively complex and in addition a deep analysis, it requires to make technical choices from the earliest steps of the development.

8.3.1 Dealing with agents

In the simulator we needed to represent humans and devices as standalone elements of the simulation. In order to do it we decided to use the JADE[17] (Java Agent DEvelopment Framework) platform which has been developed at Telecom Italia Lab which is an R&D department of Telecom Italia. This platform proposes a set of basic elements allowing developers to implement their own multi-agent projects. We chose this specific platform as it is reliable and offer all the minimum features that we wanted for the multi-agent part of our simulator: each agent has its own process and has the ability to communicate with other agents through a messaging system allowing developers to customize the management of messages. The organization of the JADE platform is quite simple: a platform is composed of a set of containers which can run on separate devices or not. Each of these containers has in

[17] JADE: http://jade.tilab.com/

charge to hold and manage its allocated set of agents. Agents can communicate within a container, within a platform and even between different platforms.

8.3.2 Dealing with ontologies

The representation of knowledge inside the simulator is obviously a critical point for us. As we have presented earlier we have decided to use ontologies to describe users' context. Given this point is has natural to use ontologies all over the application to represent knowledge in order to maintain the coherence and facilitate the manipulation of information.

Hence, our first concern was to find a way to represent ontologies. To do it we decided to rely on the Ontology Web Language (OWL[18]). OWL is a language for defining and instantiating ontologies that can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. The Web Ontology Language was adopted as the recommendation by W3C in 2004. OWL provides the required elements to represent and use complex information models. The language itself is an extension of the RDFS[19] language and provides additional features for a greater expressiveness. OWL can be used to define classes and properties and also provides constructs to create new class descriptions as logical combinations (intersections, unions, or complements) of other classes, define cardinality restrictions on properties and so on.

Our second concern about ontologies was to find a simple way to create and manipulate them easily in our application. To solve this issue we were able to use the Protégé[20] framework which offers a graphical interface to create ontologies. In addition to this interface it also features a complete Java API allowing us to easily and efficiently deal with basic ontologies elements. As we'll see in the next sections, the protégé framework was intensively used in our different Abstraction modules and helped us fasten the development of the application.

8.3.3 Dealing with Reasoning Rules

Even if manipulating ontologies was a major part of our work in the simulator, we also needed the deal with reasoning rules. As we wanted to be able to reason over ontologies and given the fact that we already had chosen the Ontology Web Language to deal with these, it was almost natural to lend toward the Semantic Web Rules Language (SWRL[21]) to represent reasoning rules. SWRL is a rule language based on a combination of OWL DL and OWL Lite sublanguages with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language (RuleML[22]). This language proposes a specification of rules in the form of implication. Thus, each rule is composed of two parts:

- an *Antecedent*: composed of a set of atoms to be evaluated;
- a *Consequent*: composed of a set of atoms to be applied if all atoms of the Antecedent are realized.

Atoms of both the antecedent and consequent can be elements of the OWL language (classes, properties, datatypes, individuals, etc.), literal values or built-ins. This last category of atoms provides some specific functions to help users and developers to write more complex and effective rules. Basic built-ins are present in the SWRL specification, but anyone can develop its own built-ins and integrate them into reasoning rules. To facilitate the

[18] OWL: http://www.w3.org/TR/owl-features/

[19] RDFS: http://www.w3.org/TR/rdf-schema/

[20] Protégé: http://protege.stanford.edu/

[21] SWRL: http://www.w3.org/Submission/SWRL/

[22] RuleML: http://ruleml.org/papers/tutorial-ruleml-20050513.html

development of such rules the protégé framework contains SWRL editing and running capabilities.

Until now we have seen what tools and languages we have used to model and manipulate OWL and SWRL but we needed another tool to apply reasoning rules on the ontology, a rule engine. For this part of the simulator we decided to use the Jess[23] Rule Engine which is powerful, efficient and lightweight. Jess was an even more natural choice as we previously had a good experience with it on another research **[Hamadache et al, 2007]** and the fact that it can be simply used with protégé and SWRL through the use of a simple rule engine bridge (the SWRLJessBridge).

We have reached the point where all major tools we used have been presented. We can now go deeper in the presentation of the simulator and its features.

8.4 Detailed Architecture

Now that we had a look at the general architecture of the simulator we can go deeper in the understanding of the architecture by giving more details about each component of the application. Before exploring every component of the simulator, we have to give more details about the design of the ontology and how it works within the simulator.

8.4.1 Ontological Design

As we have already presented in the precedent chapter, the PCSCW model can be considered as an ontology and an open set of devices collaboration rules. Given this simple fact, the design and the correct use of the ontology was a critical point in our development.

In addition to this aspect of the development we also figured out that embedding all data required by the simulation to be run was an interesting opportunity and could simplify later the management of simulation data.

Given this perspective we designed the ontology according to the PCSCW Model with some extensions dedicated to the simulation. The Figure 42 depicts the general design of the ontology as we developed it. The central disc represents the very core of the PCSCW Model with the three sub-models (role, task and resource) and device collaboration rules.

Around this core we added the necessary concepts to handle and represent events. This additional layer (called Event Model) contains some classes, relations and datatypes such as:

- *EventSequence* (owl:class): allows to define a sequence of events in order to be able to fire them in a specific order;
- *Event* (owl:class): the representation of an event;
- *EventAtom* (owl:class): as we'll see in the dedicated section, this class allows us to represent atomics knowledge modification of an event;
- *hasEventAtom* (owl:objectproperty): represents the link between an *Event* and an *EventAtom*;
- *eventContent* (owl:objectproperty): represents the link between an *EventAtom* and its associated modification of knowledge;
- *eventState* (owl:datatypeproperty): the state of the event, has it been triggered or not;

[23] Jess: http://www.jessrules.com/jess/index.shtml

- *eventType* (owl:datatypeproperty): the type of knowledge modification (creation, modification, suppression);
- *sequenceOrder* (owl:datatypeproperty): if part of one, the order of the event in its sequence;

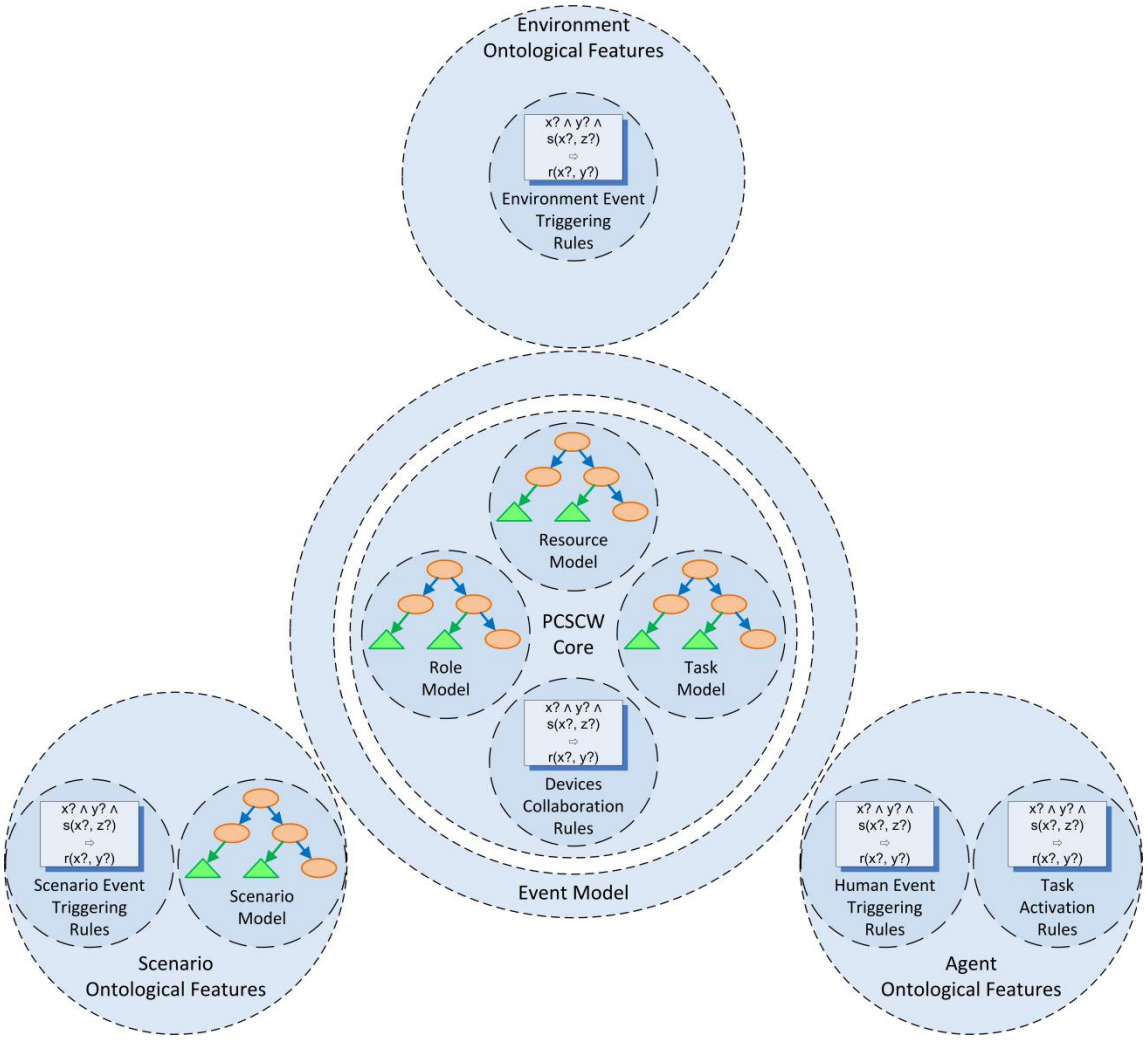

Figure 42 Ontological Design

Built over the PCSCW Core and the Event Model, we have a first another extension of the ontology and reasoning rules used by the *ScenarioEngine* module (the scenario ontological features). These features are organized as following:

- an extension of the structure of the ontology (mainly a new class called *Scenario* with its set of object and datatype properties);
- a set of scenario event triggering rules used to know when the evolution of the simulation reaches the point where a given event of the scenario has to fired.

The additions of the ontology related to the environment module are less substantial than for the scenario. Indeed, the representation of the environment is already present in the core ontology, as it can be seen as the root of all context information. In consequence for the Environment module, only some SWRL Rules have been added to trigger environment events.

Finally we reach the ontology specificities of agents. As agents simulate real devices that would use the ontology to represent their context and given the fact that there isn't any further data required to help the agent working, there is no need for an addition to the ontology structure. However, as the agents we simulate have to be able to properly react to

the evolution of their environment we needed some reasoning capacities. In this perspective we added the possibility for all kind of agents (humans and devices) to evaluate the fact that a task has to be activated or deactivated (according to the current state of their context). In a second consideration we needed to give some "intelligence" to human agents. To realize this we added (only to human agents) a set of "human event triggering rules", allowing them to fire events when the scenario unfold and the simulation evolve.

8.4.2 Simulator

The first module we need to look closer is the simulator module. As depicted on Figure 43 this module is designed around a central component representing the controller of the PAC model. This component extends an abstract class *OWLAbstractPCSCWController* that provides basic features to control an Ontology-based knowledge. Around this main component we can find five components:

- The SimulatorGUI is the Presentation associated with the simulator, in its current implementation it is used to display the basic window of the application with it main menus;
- The GUIThread is a simple class allowing us to run the SimulatorGUI in a separate thread;
- The SimulatorKnowledgeManager represents the Abstraction part of the Simulator module. It extends an abstract class called *PCSCWOntologyCore* which offers the essential features to manipulate the underlying PCSCW model and its associated sub models;
- The SimulationReporter is a subcomponent of the simulator that manages the whole process of reporting the simulation in a specific format;
- The Event Layer, as it will be discussed further in its own section is a component shared by most modules of the application in order to manage events of the simulation.

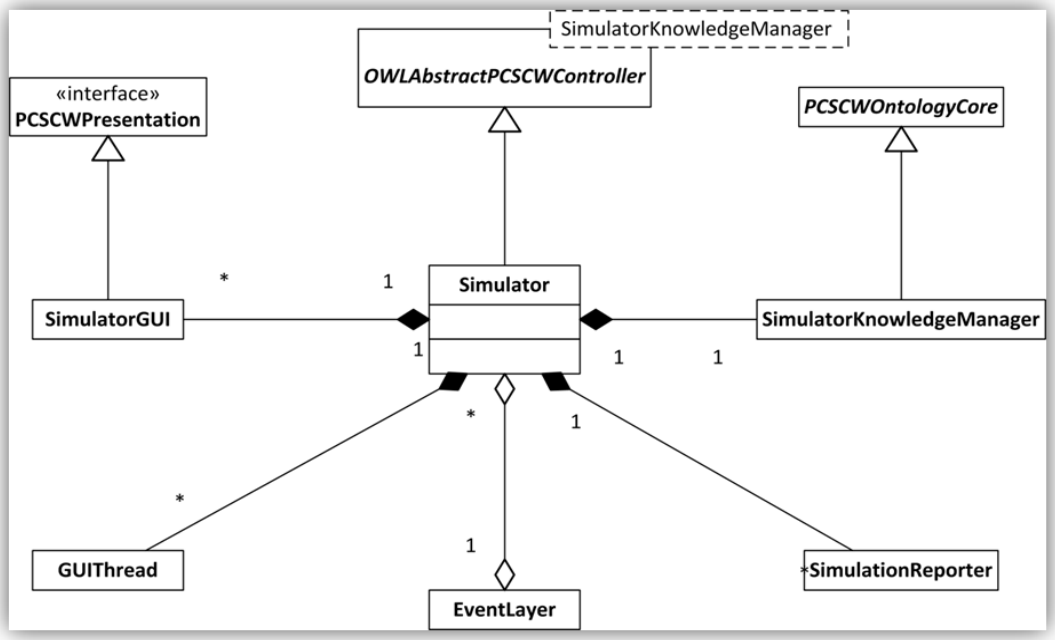

Figure 43 Simulator Module

Even if this module is an essential one, it isn't one of the most complexes. Indeed, it only has to deal with the general interface, the repository of scenario and channelling events to the simulation reporters.

8.4.3 Scenario Engine

Going one step down in the hierarchical architecture of the simulator we find the Scenario Engine. This module is detailed on the Figure 44. Let's have a look at each of its components:

- ScenarioManager: as for the simulator, this central component is the controller of the module; it has in charge to manage all other modules and allows them to communicate indirectly. As we have evoked earlier this specific component can communicate with the simulator;
- Simulator: to respect the PAC pattern the Scenario Engine is linked through its ScenarioManager to the controller of the upper module, the Simulator;
- EnvironmentManager: this module is the controller of the Environment. It is linked to the ScenarioManager as the environment can be considered as a module under the authority of the Scenario Engine, even if the Environment has its own independence;
- ScenarioControlFrame: this simple presentation is dedicated to the control of the scenario, it allows the user to load, run, pause and stop a given scenario. It obviously interact with the ScenarioManager by relaying him user's actions;
- EventsFrame: this element of the interface is designed to display all events of the simulation;
- MainSimulationFrame: whereas the ScenarioControlFrame controls the unfolding of the scenario, this component gives a graphical representation of the simulation.
- ScenarioKnowledgeManager: in the same way as the SimulatorKnowledgeManager is the abstraction of the simulator, this component manages knowledge of the Scenario Engine. As for other dedicated knowledge managers it provides access to specific parts of ontologies. In this case it allows to acces to all data relative to the unfolding of the scenario: the ordered set of scenario events and when they have to be fired;
- ScenarioThread: this thread is used to isolate the running of the scenario from the rest of the scenario engine. It allows the scenario to keep unfolding while performing other treatments (integrating knowledge, preparing data to be displayed…);
- RuleEngineManager: this specific component is used to reason over a given knowledge manager. Indeed, contrary to the Simulator, the Scenario Engine can directly interact with the environment and agents of the simulation by firing and dispatching them new events. To do it the Scenario Engine has to infer the fact that conditions to fire the event are met (and that it hasn't already been fired), to realize it efficiently we use a RuleEngineManager that allows us to drive the Jess rule engine;
- EventLayer: the same component as for the simulator, it allows the Scenario Engine to dispatch and intercept events from other modules of the simulation.

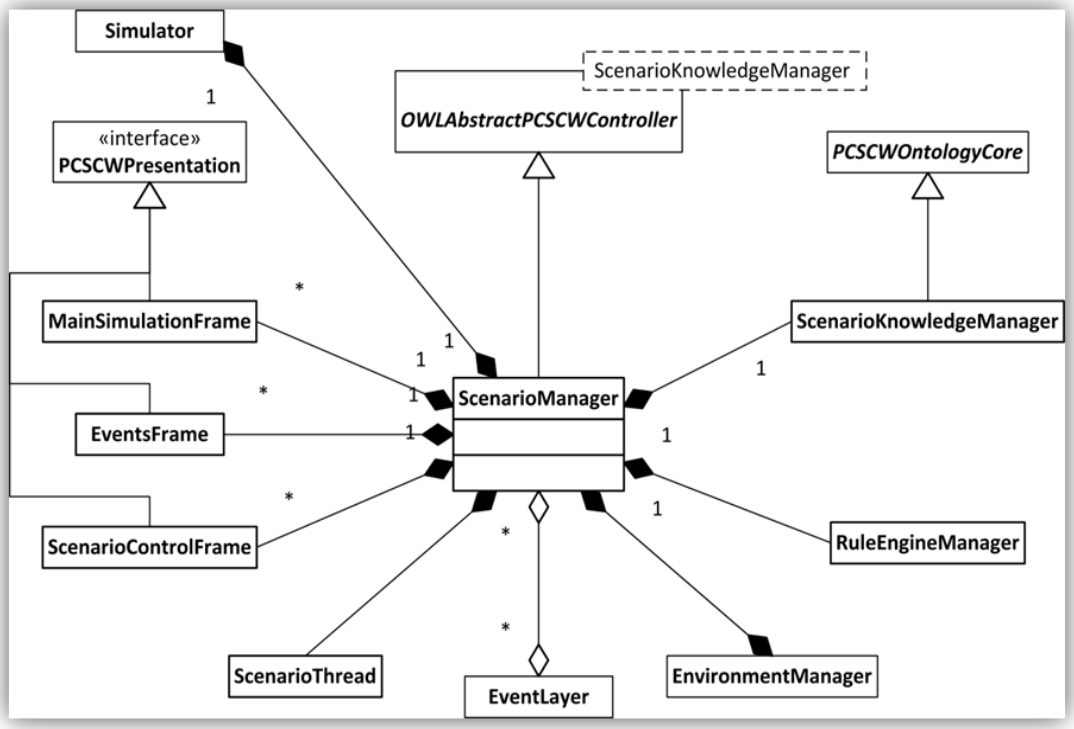

Figure 44 Scenario Engine

The Events Frame is also useful to monitor how the collaboration was managed according to the succession of events.

The MainSimulationFrame provides us a minimalistic view of the simulation; we can thus monitor the existence of agents, their interactions (messages exchanged). We can also see the evolution of collaborative tasks, completed ones, currently actives one and those that still have to be started.

8.4.4 Environment

The following module to inspect is the Environment. This module of the application has in charge to represent the environment of the simulation. This role in the simulator implies that the environment has to communicate with agents to provide them relevant data about their surrounding context; it also has to trigger events according to agents' behaviours and evolution of the scenario. As depicted by Figure 45 the environment module is composed of numerous components:

- EnvironmentManager: the controller of the module, manages all other modules and ensure the communication between them;
- ScenarioManager: as we have seen previously the Environment module communicate with the Scenario Engine through the ScenarioManager;
- EnvironmentKnowledgeManager: the Abstraction of the module, it stores and manages all information of the environment (containing all data of agents and of their "surroundings"). As for other specific knowledge managers it offers a dedicated range of method to collect and alter information related to the environment;
- RuleEngineManager: in the perspective of firing events due to the evolution of the environment, we need to have a RuleEngineManager allowing the environment to reason over its context to know which event to fire;

- jade.core.Runtime: the JADE runtime used to support the main agent container of the simulation;
- jade.core.Profile: the profile used for the creation of the main agent container of the simulation;
- jade.wrapper.ContainerController: the main agent container of the simulation, it starts and holds all JADE agents. As we'll see later a JADE agent is only a part of the representation of an agent in the simulation;
- AgentFactory: at the beginning of the simulation, all agents are created according to the description of the environment, this class is used to build them by dealing with the EnvironmentKnowledgeManager and the agent container;
- AgentsListFrame: this Presentation controlled by the EnvironmentManager presents the list of agents participating to the simulation;
- EventLayer: as well as other modules, the Environment uses the EventLayer to dispatch and intercept events;

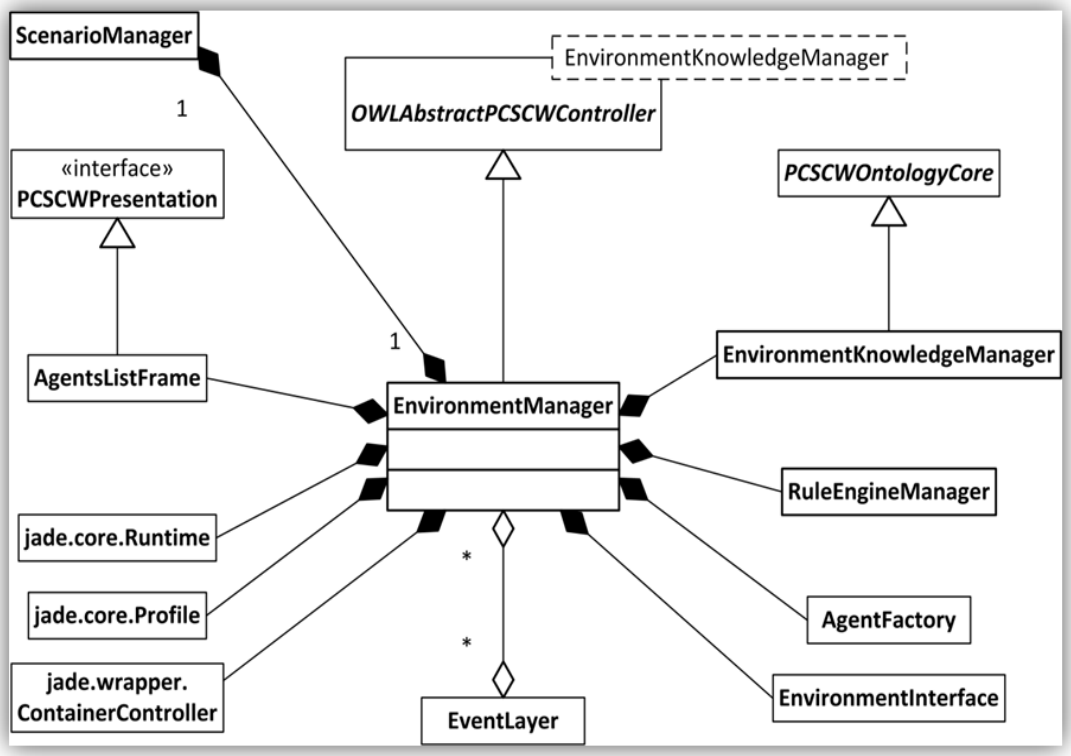

Figure 45 Environment

The role of the environment in the simulator is a major one; as we have said it has in charge to handle the creation of agents by using its knowledge to pilot the JADE container through the use of the AgentFactory. It also has to fire events according to the evolution of agents' context. Those events are essential in the unfolding of the scenario and for the simulation itself, has it can condition the correct choice of behaviours for agents.

8.4.5 Agent

Until now we've seen most of the major modules of the application, the last one to look at is probably the most important of them: the Agent module. Let's see how the agent module is build (Figure 46):

- AgentManager: obviously the main component of the module, it has in charge to control the agent by orchestrating all the other components. As most of the other

controller of the implementation of the simulator, it inherits from the abstract class OWLAbstractPCSCWController, giving it essential features to manipulate events;

- AgentConstants: this interface is used as a set of constants shared by all implementation of agents;
- AgentKnowledgeManager: once again this specific knowledge manager provides to the agent a set of method allowing it to access and manipulate its own knowledge;
- RuleEngineManager: the role of this RuleEngineManager is completely different from those of the other modules. Indeed in the agent module it is used to reason over the agent knowledge and find the best way for the agent to collaborate with others by using the device collaboration rules of the PCSCW model;
- PCSCWJadeAgent: this component is the part of the agent module that deals with the JADE platform. It relies on the extension of the jade.core.Agent class which is used to represent JADE agent in the JADE container;
- BehaviourEngine: the very heart of the "intelligence" of the agent, this component obviously deals with agent's behaviour. It is the one component that tells when to look for a device collaboration, when to reason on the context, what to do when an agent sends a message;
- AgentListener: this interface defines a set of method the agent calls when it perform some specific tasks (looking for device collaborations, sending a message to another agent, firing an event…..). Any component wishing to listen to the agent has to implement this interface and register with the agent;
- EventLayer: the common layer used by other modules of the application to dispatch and intercept events;

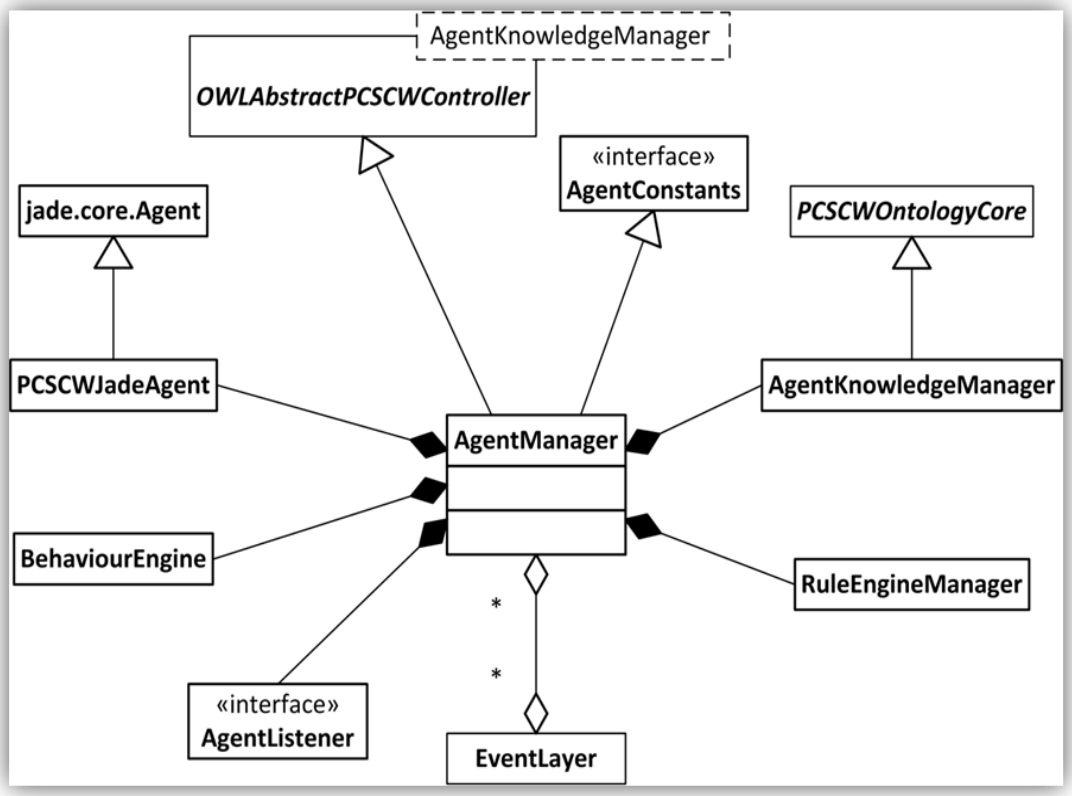

Figure 46 Agent

Unlike other modules of the application it doesn't have an upper level module as it works as a standalone part of the simulation.

As we'll see in the following sections, the PCSCWJadeAgent, as an agent of the JADE platform, allows Agents modules to exchange messages and then channel the communication between them. We'll see that some specific features have been developed to properly handle the communication and the coordination of agents.

8.4.6 Events

This section we will present and detail how events are represented, fired and dispatched throughout the system. It is critically important for the general architecture of the simulator, as it ensure the communication of context and knowledge modifications to the related modules.

8.4.6.1 Event Design

Before looking at the details of the Event Subsystem Architecture we have to present how events have been designed to be used by any modules of the system and potentially any other module we'd like to add in the future.

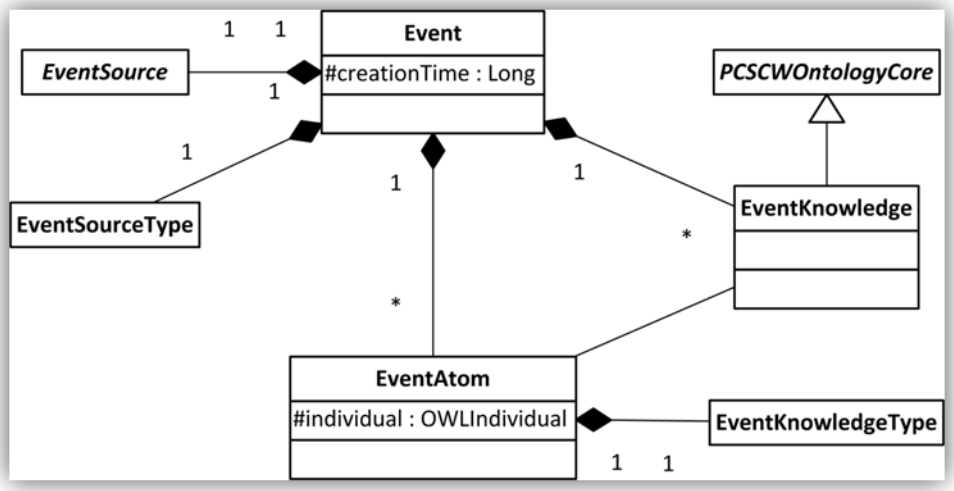

Figure 47 Event

On the figure above (Figure 47) we have the description of how an Event is represented inside the application. An Event has a *creationTime*, which is used as its ID through the process of the different modules. In addition to its ID, an event has a reference to the module that fired him, the *EventSource*. In addition to this reference, the event also has a mention of the type of the event source: an instance of the *EventSourceType* enum.

The content of the event itself has been inspired by the rest of the application. Hence, this content is nothing else than a part of the knowledge of the simulation. Given this consideration it was almost natural for us to represent the knowledge of each event by a small ontology. To do it we created the *EventKnowledge* class that extends the PCSCWOntologyCore, providing it all the basic method to access and manipulate the base of the PCSCW Ontological model. Thus, every event fired in the application has its own small ontology representing the actual fact of the event. However this instance of EventKnowledge isn't sufficient for modules of the application to properly consider and process the event. Indeed, an *Event* as we consider it inside the simulator is a set of atomic modifications of the ontology (and then the knowledge of the event) that have to be processed together in order to avoid any incoherence while handling the knowledge of the event. To help process each of these atomic modifications we have created the *EventAtom* class which contains a reference to an OWLIndividual (the representation of an Individual in an ontology according to the

Protégé-OWL API). The referenced Individual (stored in the *EventKnowledge*) represents the atomic modification of the knowledge. To complete the design of events, the atoms (EventAtoms) contains a precision on the type of knowledge referenced by the atom (addition, modification of suppression in the ontology).

Thus, if we'd like to summarize the design of events we could say that an *Event* is a set of *EventAtoms* referencing the atomic modifications of knowledge contained in the *EventKnowledge*.

Now that all basic modules of the prefigured architecture and the event design have been presented we have to get a closer look (Figure 48) at the event architecture. As it is used by most of the controllers of the application, it can be considered as a subsystem.

8.4.6.2 Events Subsystem Architecture

The main component of this part of the application is the EventLayer. This component, shared by all modules of the simulator, provides them the ability to construct and dispatch events. In order to effectively handle its duty this component uses several sub-components:

- EventFactory: this component is used by the event layer and other modules to build an event. Events are stored (in the *eventCreationMap*) according to their creation time which is used as their ID;
- EventNotificationRegistry: as modules can re-dispatch events toward other ones it is necessary for us to ensure that a given module hasn't already received an event in order to avoid any re-processing. In this perspective, this class stores a mapping (*eventsNotifyedToListeners* and *listenersNotifyedOfEvents*) between events and listeners that have been notified.

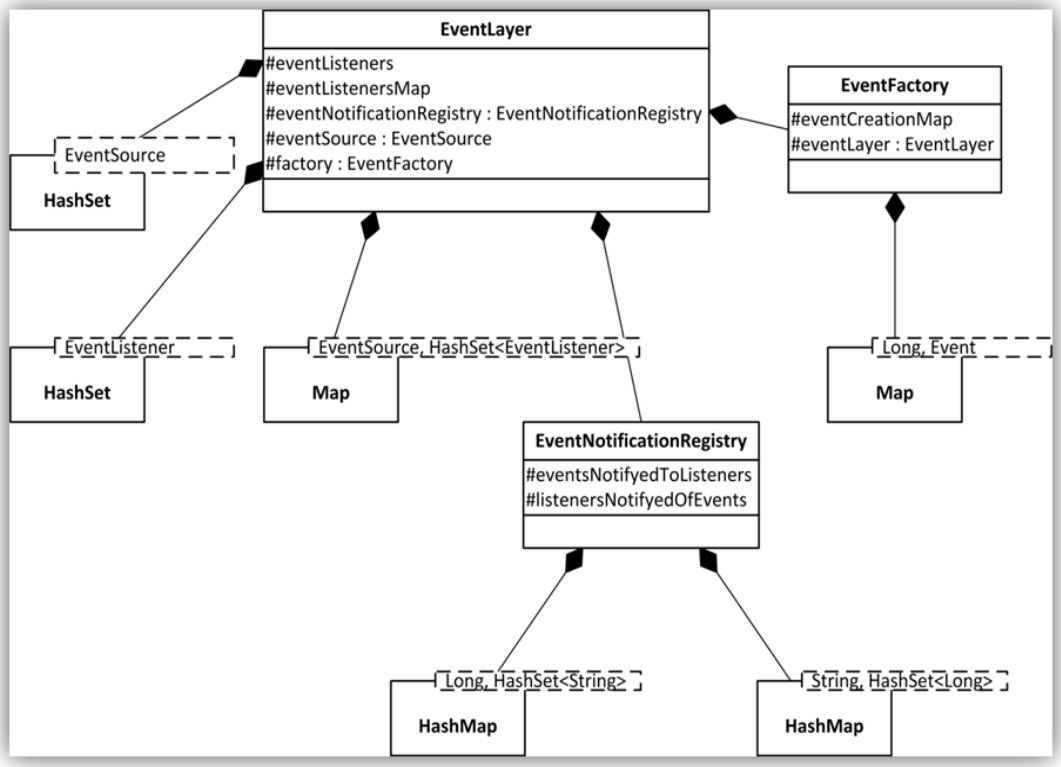

Figure 48 Event Architecture

From the Figure 48 it is noticeable that the *EventLayer* and its subcomponents are manipulating the *EventSource* and *EventListener* concepts. These two concepts simply

correspond to components that fire events (*EventSource*) and components that intercept events (*EventListener*).

By using the two components we previously detailed and allowing controllers to register to be notified of events, the *EventLayer* provides a common way for all modules to dispatch information to others.

The Figure 49 illustrates the point we've just made by showing how an event is dispatched to a given set of modules according to the *EventListeners* related with an *EventSource*. More precisely here, the "Human Agent 1" (HA1) is an *Agent* of the simulation who wants to send an event to all modules listening to him. Thus, our "Human Agent 1" is the *EventSource*, dealing with the *EventLayer* to actually build its event in a first time (by collecting and integrating all *EventAtoms* within the *EventKnowledge* through the use of the *EventFactory*) and fire it in a second time. Once the event has been fired, the *EventLayer* handles it and, by looking both at its internal source/listeners map and its EventNotificationRegistry, is able to correctly dispatch the event to all EventListeners that had previously registered to receive events from HA1. As depicted on the figure, other modules of the application may not have registered to receive events from our agent. In this specific example the "Human Agent 2" (HA2), the *Simulator* and *Scenario Engine* modules don't receive events from HA1.

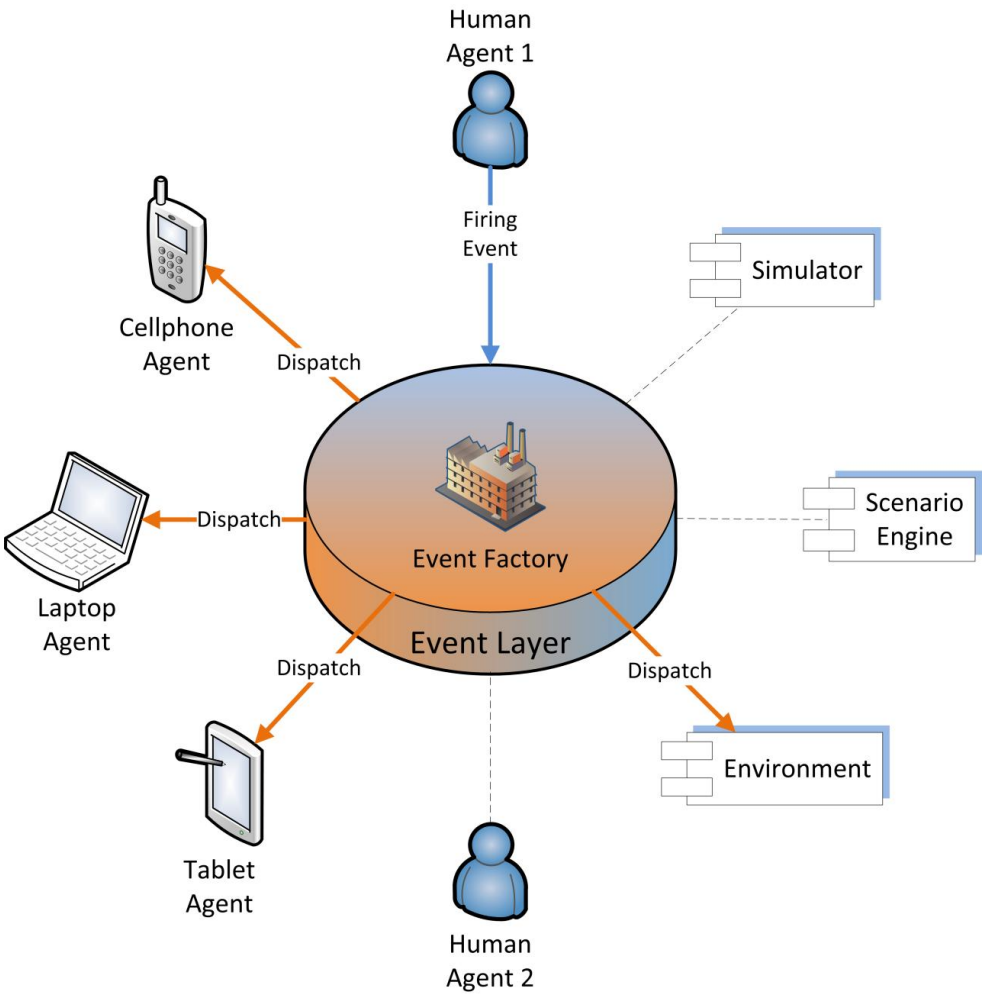

Figure 49 Event Firing and Dispatching

The events architecture can seem pretty simple but it effectively allows any module of the application to communicate and evolution of its knowledge to the precise set of interested modules. This mechanism also provides an effective way for external modules to collect or send events to any other module of the simulator. For instance if we wanted to have an

external module for collecting all events of the application and trace them in an XML document for later analysis we could simply add a dedicated module listening to all events of the simulation.

8.5 Architecture Mechanisms

As the general architecture of the application and most of its modules have been more deeply presented, we can now have a good overview of how the simulator actually works. To complete the description of the simulator we need to introduce some other internal mechanisms that have been designed to make effective the simulation.

8.5.1 Agents Communication

The first internal mechanism we have to introduce here is the communication between agents. Indeed, until now we have presented that the different modules of the application can interact by notifying other module that their context has evolved through the firing of events. In the simulation perspective though, we also need to make agents communicate as if they were communicating over a network (for device agents). As we already evoked in the previous section, this need was the initial motivator for the use of the JADE platform as it help developers design communicating multi-agent system. Given this opportunity we have developed our own use of the JADE platform to properly handle the communication of agents and the coordination of exchanged messages.

8.5.1.1 Agent Communication Structure

The very heart of agents of the JADE platform is the set of behaviours is using. Indeed, each agent of the platform has a given set of object called Behaviours that describe the functioning of the agent according to its current state. All active behaviours of an agent are called on each "tick" of the agent.

In this perspective, JADE offers different kinds of simple behaviours:

- Cyclic behaviours: behaviours which stay active as long as the agent is alive, this kind of behaviour can be used for repetitive tasks;
- Ticker behaviours: a subtype of cyclic behaviours which are triggered periodically;
- One shot behaviours: this behaviours are only executed once and are "destroyed" afterwards;
- Waker behaviours: those one-shot behaviours are executed once at a specific time;
- Receiver behaviours: behaviours triggered when the agent receives a given type of message.

In addition to this non-exhaustive set of simple behaviours, the JADE platform also offers an interesting range of composite behaviours (that is to say behaviours composed of several sub-behaviours):

- Parallel behaviours: this kind of behaviour controls a set of sub-behaviours running in parallel without any specific order;
- Sequential behaviours: contrary to the precedent kind, this one forces its sub-behaviours to run one after another.

It is obviously possible to design composite behaviours as any of the previous simple kind (a sequential behaviour can also be a WakerBehaviour). To properly make what we intended we had to use the behaviours mechanisms and design ours own.

The following figure (Figure 50) depicts the use and adaptation of the JADE behaviours and messaging features we made.

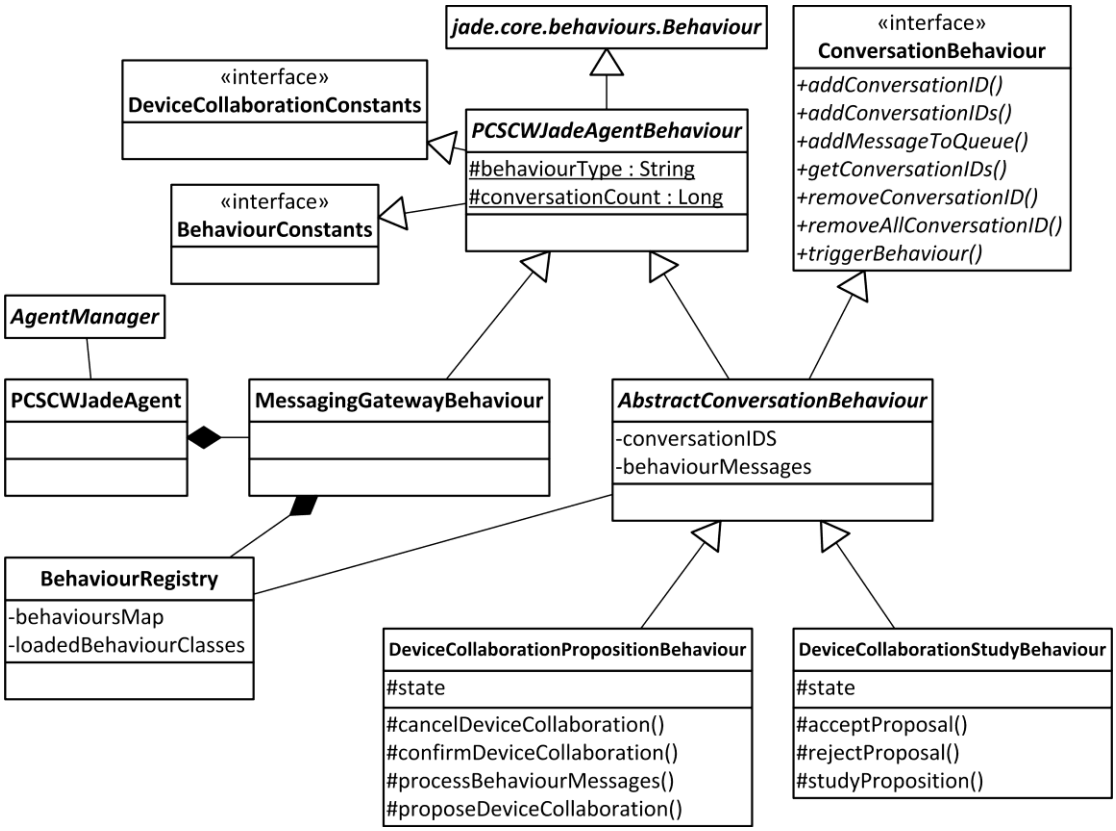

Figure 50 Agents Communication Architecture

The *PCSCWJadeAgentBehaviour* is an abstract class that extends the Behaviour class of the jade.core.behaviours package. It is intended to be the mother of all other behaviours we use in the simulator. From the class of the JADE platform we added a behaviourType as a static protected member of the class. This field allows us to define a type for each subclass of *PCSCWJadeAgentBehaviour*; as we'll see in the following subsections this field allows us to know if a given type of behaviour is already running for an agent. We also added another protected static member to the class: the conversationCount. It will be used by all classes to increment the number of conversations and generate a unique ID for each new conversation. *The PCSCWJadeAgentBehaviour* class also implements the *DeviceCollaborationConstants* and *BehaviourConstants* which are interface used to define static fields such as topics of conversation, types of behaviours, etc.

Beneath this first class we've got two main subclasses:

- *MessagingGatewayBehaviour*: this behaviour is the one in charge of reading and "dispatching" incoming messages of the agent to its other behaviours;
- *AbstractConversationBehaviour*: this abstract class is the mother of all other behaviours dealing with at least one conversation. It completes the upper class by adding a set of conversationIDs representing the current conversations associated with this behaviour. It also had an ordered list of behaviour messages sorted according to their reception time. In addition it implements the *ConversationBehaviour* interface which defines a set of methods to manipulate conversations.

As we have said the *MessagingGatewayBehaviour* is used to intercept all messages of the agent and re-dispatch them to their related behaviours. In order to correctly manage this feature this class uses a *BehaviourRegistry* that keeps mappings between conversation IDs and their associated behaviours. The messaging gateway is also the only behaviour "known" by the agent as all other behaviours are managed by this one.

8.5.1.2 Communication & Behaviours

For now we have seen the structure of the implementation of the JADE behaviours and messaging for our application. We still to see this structure works to make agents actually communicate and coordinate themselves.

The Figure 51 illustrates how the different elements of the architecture work together to channel and support the communication of agents.

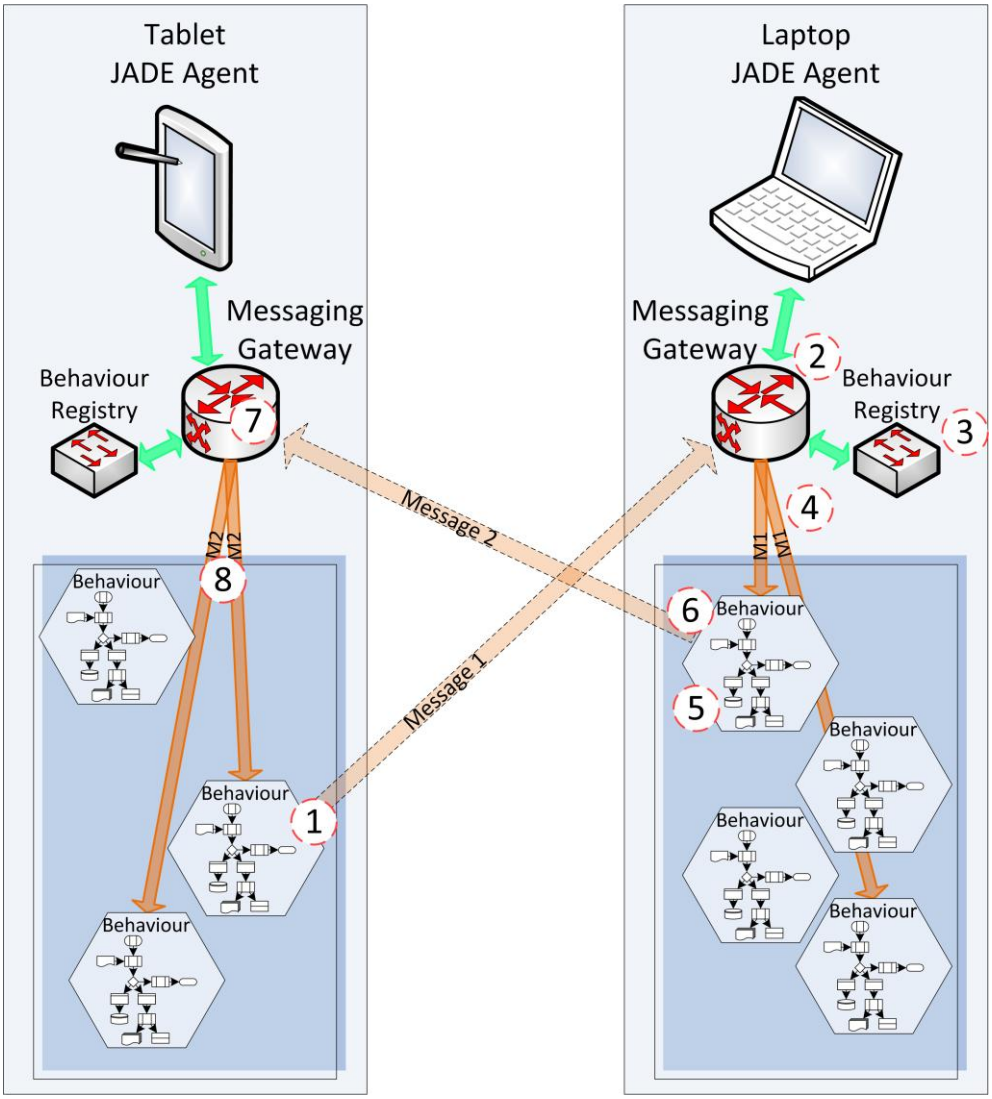

Figure 51 Behaviours and Messages

On this figure we've got the example of two JADE agents (in fact *PCSCWJadeAgents* in our simulator) exchanging two messages, one in response of the first. Let's see how the communication takes place:

0. The agents "*Tablet*" and "*Laptop*" both have *MessagingGatewayBehaviours* running with their associated *BehavioursRegistrys* and a set of other behaviours currently actives;
1. One of the conversation behaviours (BT1) of *Tablet* sends a message to *Laptop* with a specific Topic and conversationID;
2. The messaging gateway of *Laptop* reads the message and consults its BehaviourRegistry to know if there is any behaviour that matches the conversationID.

3. In addition the gateway asks the BehaviourRegistry for all Behaviour classes loaded in the runtime (except for those which have an instance already running) and evaluate each one of them to know if an instance has to be created and run. This mechanism makes *Behaviours* work as "plugins" that can be added to simulator at any time. This could be done even while running the simulation with a simple addition to the existing *BehaviourRegistry* that would check periodically a given set of *Behaviours* repositories for new ones.
4. Two behaviours matching the *conversationID* are found and the messaging gateway put the message on both their *behaviourMessages* queues.
5. The first concerned behaviour (BL1) picks the message, reads it and processes its content.
6. BL1 then sends a message back to *Tablet* with the same conversationID and Topic.
7. The messaging gateway of *Tablet* then asks its registry for matching behaviours.
8. The message is then sent to the concerned behaviours (among which we've got BT1).

With this simple and effective mechanism we can make agents communicate and have "conversation" with complex and extendable patterns. If we give a look back to the Figure 50 we can see that there are two more classes we have not presented: the *DeviceCollaborationPropositionBehaviour* and the *DeviceCollaborationStudyBehaviour* which both extends the *AbstractConversationBehaviour*. At the light of what we just presented on the way behaviours works together, their functioning becomes almost obvious:

- The *DeviceCollaborationPropositionBehaviour* is designed to propose a Device Collaboration to other agents and manage their responses. If all agents related to this proposition accepts its, then this behaviour sends them back a confirmation and the collaboration can take place;
- The *DeviceCollaborationStudyBehaviour* works with the *DeviceCollaboration Proposition* as it is the behaviour used by agents to analyse the device collaboration proposition sent by another agent and accept or decline it.

Thus these two, relatively simple, classes allows agents to propose and accept or decline devices collaboration propositions, coming straight from the reasoning made on their knowledge ontologies.

8.5.2 Event Triggering

The last mechanism we have to present is the way how events are triggered by the different modules of the application. Indeed until now we have seen how the Event Subsystem Architecture is designed to properly fire and dispatch events; however we haven't seen how these events are triggered before being dispatched.

In order to explain it a little deeper we've taken the example of human events. This kind of events is only triggered by Human Agents. On the Figure 52 we represented how an interaction with a human agent can trigger the firing of a Human Event.

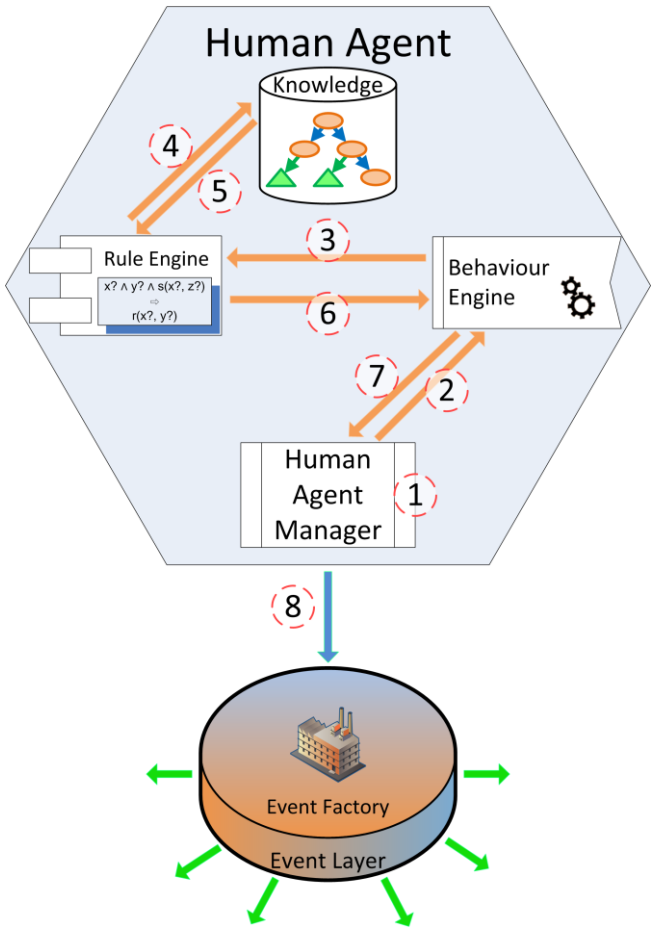

Figure 52 Human Event Triggering

Let's have a look at how the different parts of the Human Agent Module are organized and work together to effectively fire a Human Event when it's required:

1. The Human Agent Manager is notified of an event (that is to say a change in its current knowledge);
2. The agent manager notify the Behaviour Engine of the event;
3. Among the other tasks (knowledge integration, tasks activation…) performed by the Behaviour Engine on reception of this new event, it also needs to know if a new Human Event has to be triggered. To do so, it simply asks the Rule Engine (even if it is represented as a direct link on the figure, as the behavior engine and the rule engine are two direct subcomponents of the Human Agent modules, their interaction is channel by the human agent manager);
4. Then the rule engine manager loads all the required knowledge and reasoning rules related to the triggering of events in the rule engine;
5. Once done it can reason on the context and find all events that have to be triggered in the present state of agent's knowledge;
6. The rule engine manager returns the set of Human Events found to the behavior engine;
7. Then the behavior engine tells the agent manager to fire these events;
8. With the help of the Event Layer (and Event Factory) the human agent manager can then fire and dispatch the human events.

As we have seen the process to trigger a human event isn't really complex but involves most of the subcomponent of agent's modules. The mechanisms used to trigger Environment and Scenario Events are quite similar to this one. There are two major differences with the one we just presented. The first one is that Environment and Scenario modules don't contain a Behaviour Engine, and in consequence drive the rule engine manager directly through their module manager. The second difference is the fact that the rules to trigger human, environmental and scenaristic events aren't the same, which induces a small difference in the rule engine manager. For the rest of the mechanism (reasoning, firing and dispatching), the process is identical.

8.6 Use of the simulator

The use of our simulator has been designed to be as simple as possible. We will now present its general use

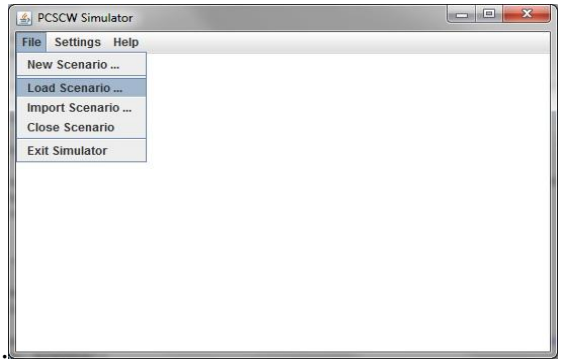

Figure 53 Simulator Main Menu

The Figure 53 shows the main menu of the simulator which allows us to load, import and close a scenario. It has been planned to integrate a scenario editor inside the simulator but hasn't been implemented yet. (for the moment, the scenarios are designed with the use of Protégé).

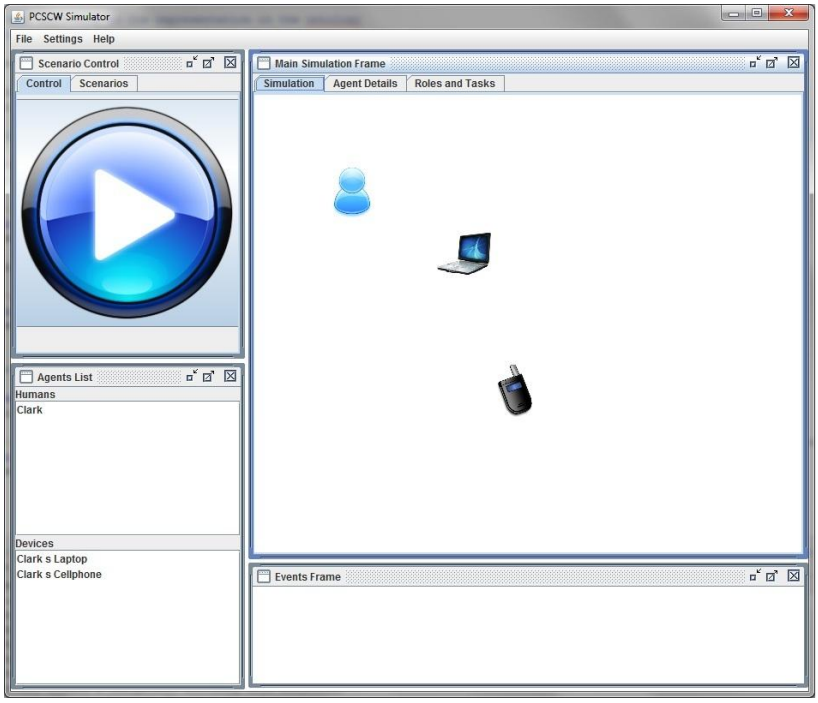

Figure 54 Simulation start

- Once a scenario repository has been selected we can select and load the desired scenario. On the Figure 54 we've got the simulator just after the end of the scenario loading. As we can see the interface is organized in four internal frames: On the top left we have the Scenario Control Frame which allows us to select, load and control the start and the pause of scenario;
- On the bottom left we have the Agents List displaying the list of agents taking part to the simulation divided along with their kind (human or device);
- On the bottom right the Events Frame displays all the events of the simulation;
- On the top right, the Main Simulation Frame allows to monitor the evolution of the simulation by representing the different agents (with icons) and their interactions. In addition, the different tabs of this frame allow to consultation of resources, agents, tasks and roles details.

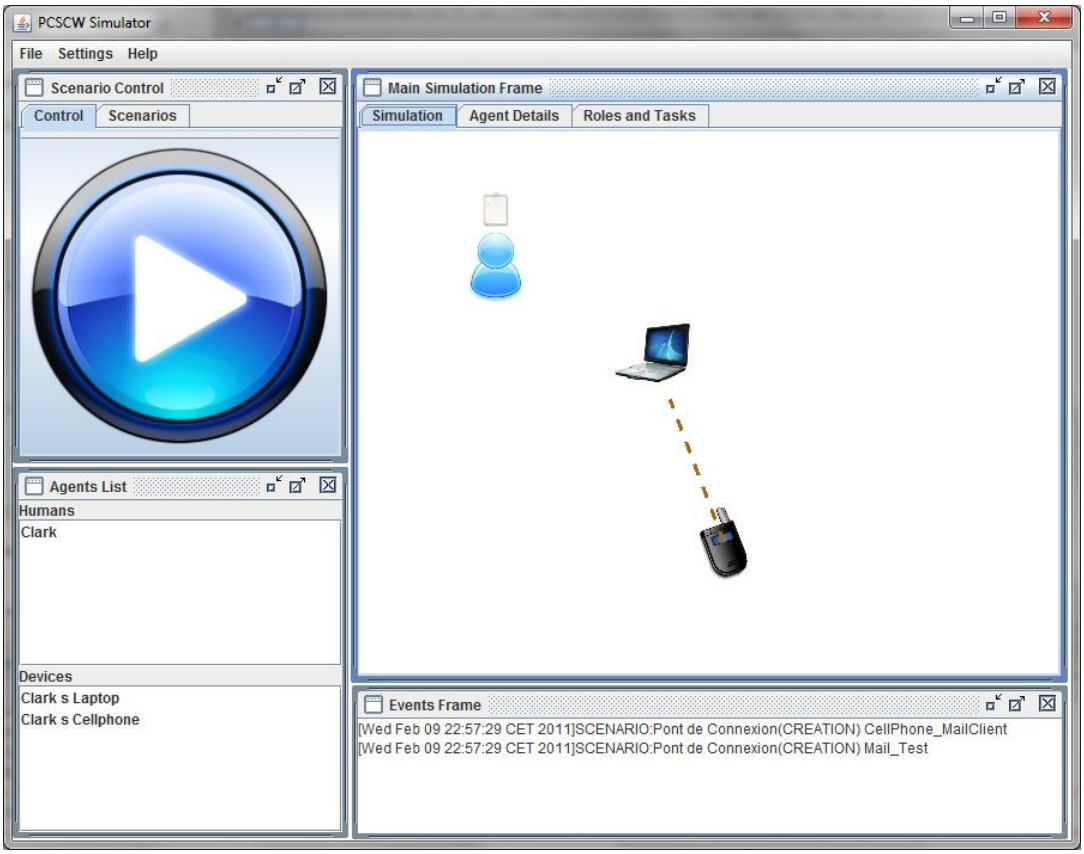

Figure 55 Simulation Running

The Figure 55 is a screenshot of the simulator playing a scenario. On this picture we've got three agents represented: a human agent (Clark) and two device agents (Clark's Cellphone and Laptop). The little icon over the head of Clark indicates that he has an active task he has to complete. The reddish dotted line between the two devices depicts the fact that they are currently exchanging a message. On the event frame we can notice that the simulation has already known 2 events.

The Figure 56 shows what happens when the scenario ends. A popup box appears and informs the users that the scenario has been completed and indicates the time it took to end. If any scenario reporter has been configured to report the simulation it can give further indications (for example where to find the final report). The user can finally consult the report according to what type of report had been configured.

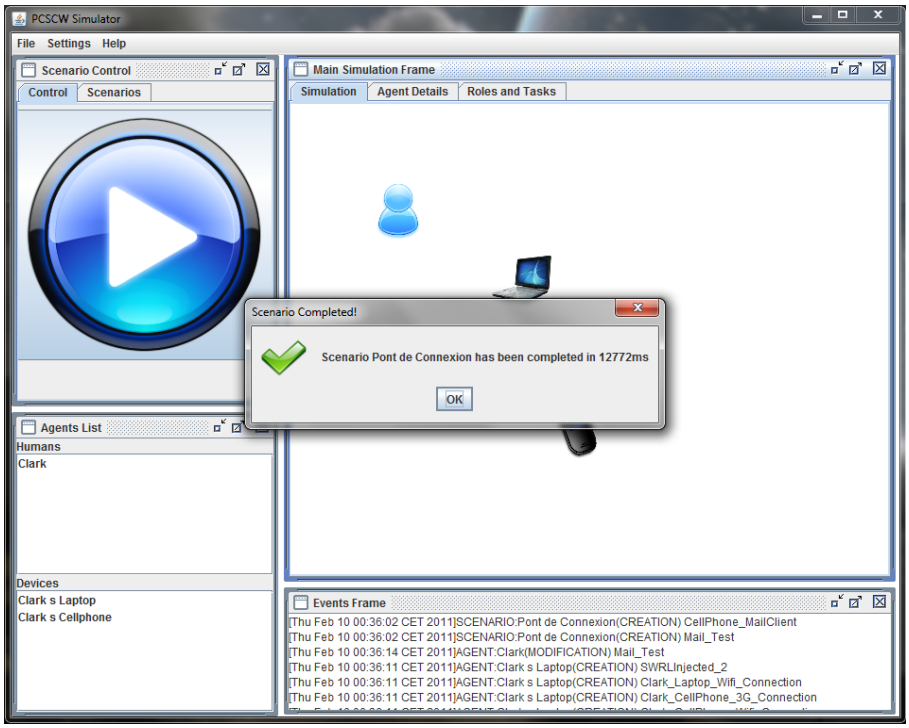

Figure 56 Scenario Completed

As depicted by Figure 57 the user has the opportunity, once the simulation is completed, to directly consult the details of the simulation on the simulator. This specific interface provides to the user the possibility to see what happened during the simulation: how long did the simulation take, what actors were involved, what were the roles, tasks and actions of agents. It also presents details of the different collaborations that happened between devices and what messages were exchanged to allow these collaborations.

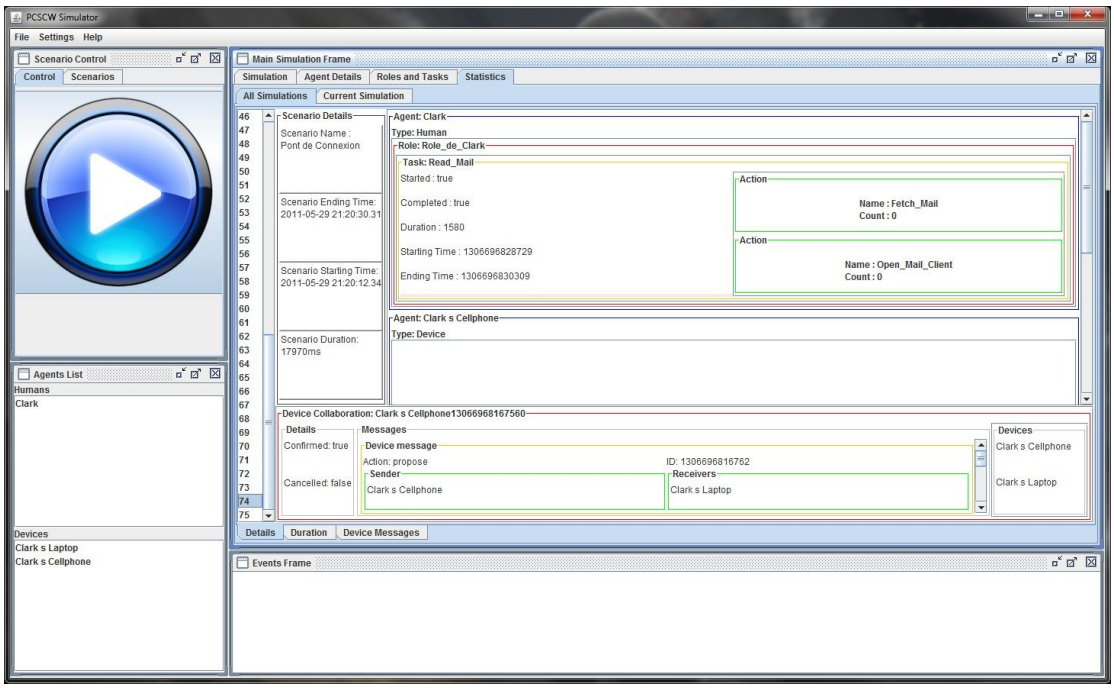

Figure 57 Finished Simulation Details

In addition to the details of the simulation itself, it is also possible to consult the statistics associated with the simulation. Statistics displayed can be relative only to the simulation itself, or to a set of simulations. Figure 58 shows, for instance, the evolution of

scenario duration according to successive simulations. This kind of synthetic view can also be used to see the evolution of tasks durations, the number of actions performed, the number of messages exchanges by devices and other useful indicators.

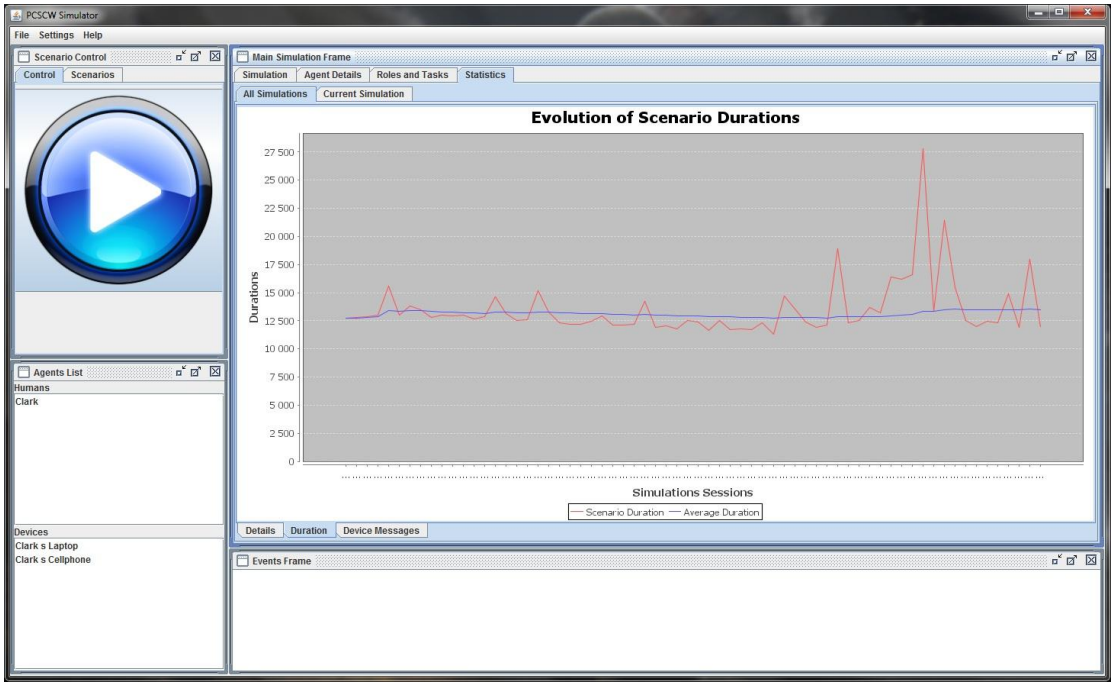

Figure 58 Evolution of Scenario Duration Statistics

Those indicators can help device collaboration rules designers to evaluate their work by making numerous simulations and simply consult the resulting statistics.

8.7 Synthesis

We developed this Simulator in the perspective of offering the PCSCW model its own simulation tool allowing us to evaluate it (as we'll see in the following chapter) and to help the design of device collaboration rules. Starting the development of such application wasn't a simple decision, we had limited time, limited resources and without great experience of developments for a realisation that seemed really complex. In the end the tool we have developed isn't completely what we expected, we had to make compromises in order to be able to complete the rest of our work in a decent time and we know that there are still many points that have to be developed or at least improved. However, and as the next chapter will perfectly illustrate, our simulator works and has given us interesting and useful results about our model.

Chapter 9. Evaluation of PCSCW Model

For a domain as young as the Computer Supported Collaborative Work in a Pervasive Environment it is essential and mandatory to explore and find as much new theories and models as possible in order not to miss anything. However, if you plan to effectively support your model it is absolutely necessary for you to evaluate what you propose. Indeed, not only as we think that our model can be an interesting and valuable contribution to this domain, we also want to make sure that our model can actually work and be efficient. Despite this absolute need we have to be honest and face the reality; evaluating such kind of system is quite a challenge from many aspects; it is highly time consuming, costly, the extreme number of parameters is hard to handle and the situations we have to create can be relatively tricky to realize and analyse. As we'll see in the development of this chapter we have decided to base a part of our evaluation on the use of our simulator, which has offered us significant advantages over evaluation in real conditions.

9.1 Evaluation Protocol

Before looking at the value of our model we have to determine how it will be evaluated. In this perspective we have to define our specific approach to the evaluation of this model, in a theoretical perspective and how it will be performed technically and logistically.

9.1.1 Evaluation Approach

As we told in introduction of this chapter, evaluating such kind of system as ours is never a nice cruise in the Cyclades, it more often seems like a round-the-world yacht race. Indeed evaluating a collaborative and pervasive system implies that we have to not only evaluate how humans are interacting between and how their system are supporting their respective owners to do their tasks, but it also require to evaluate how devices (and probably even devices that weren't directly involved in the collaboration at first) are collaborating to properly channel the collaboration of users.

Thus we have to consider three evaluation aspects: human collaboration, device collaboration and finally how device collaboration enables and channels human collaboration. In the perspective of evaluating these aspects we can't limit our evaluation to isolated unit tests but we have to build more complex sets. On the other hand, the nature of our model implies the need of a sustained evaluation over several collaboration phases. From these needs it becomes quite clear that the relevant evaluation artefact for us will consist in collaboration scenarios implying several devices and humans.

Then, the main principle of our evaluation protocol will be to put a set of humans with their surrounding environment and electronic devices in a specific situation and see if and how they can collaborate to fulfil their tasks. Once this statement has been made it becomes natural and obvious for us to design our evaluation sets as collaboration scenarios. As we said, these scenarios will put a set of humans and electronic devices in a precise situation and see how the collaboration can take place. In addition to the initial situation, they also contain a set of scenaristic events altering the situation at some precise points of the collaboration. To finally formalize a scenario used in our evaluation, we can refer to the following figure (Figure 59).

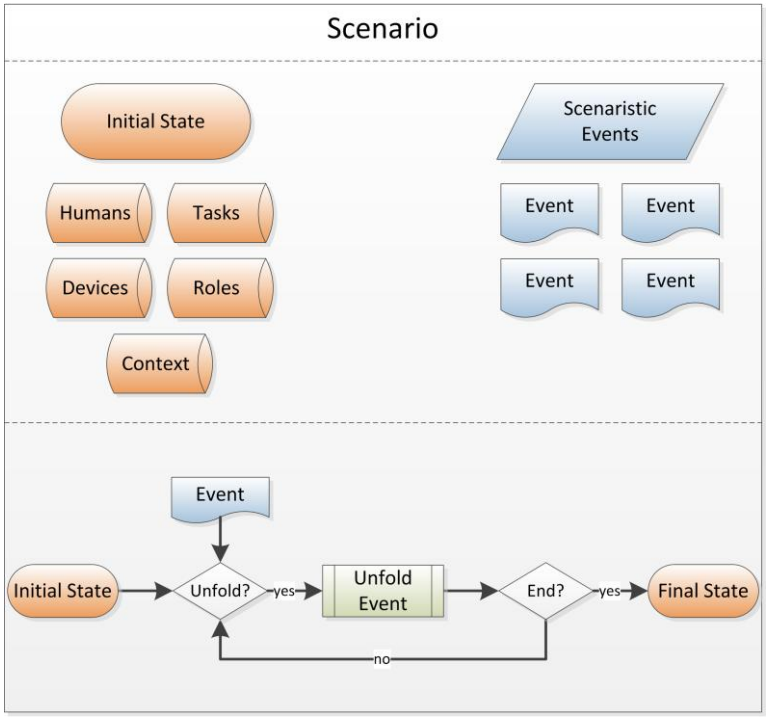

Figure 59 Use of Scenario

As depicted by the upper part of this figure, a scenario used for evaluation is composed of an *Initial State* containing the set of humans, devices (as evoked previously), but also the description of the context in which the scenario takes place (i.e. all environmental data that may be useful in the scenario). In addition it also includes the set of roles played by humans and the set of tasks they have to perform.

The lower part of this figure presents how a scenario is articulated. From the initial state of the scenario, we have the set of scenaristic events which have to be fired according to the evolution of the scenario. The representation of these events contains the requirements for them to be triggered. Hence the unfolding of the scenario is the following:

1. The initial state of the scenario has to be set up;
2. The scenario begins;
3. When the requirements are met, the next event of the scenario is unfolded;
4. The last event of the scenario is fired; it marks the end of the scenario and then the *Final State* of the scenario.

We have defined what will support our evaluation. However we don't really have described how this evaluation will be done. The optimal way to evaluate such kind of proposition is naturally to play the corpus of scenario with and without our system. Then we need to compare how the scenarios have unfolded in both cases and what are the differences between the two situations.

A particular aspect on which we'd like to insist is the fact that our model bring a new way to deal with the collaboration of human. In this perspective, even if its implications and contributions are both qualitative (it may avoid the user to do repetitive or meaningless actions) and quantitative (it can reduce the time to accomplish a task); the evaluation of the model has to focus on the qualitative part. Indeed, the qualitative issues tackled by the use of the PCSCW model are most of the time an expression or a premise of a quantitative advantage. Finally, even if the consideration of quantitative gains may be interesting, the analysis of qualitative gains (and eventual drawbacks) is way more useful as they can express

new opportunities for the collaboration whereas the quantitative is bound to the enhancement of the existent.

9.1.2 Evaluation Configuration

Now that we have defined the formal protocol of evaluation we use, we need to present the logistic configuration in which it takes place. Indeed, as we have already argued, the evaluation of a model like the one we propose is a complex and expensive challenge. It poses several constraints that can be hard to deal with:

- it is highly *time consuming*: as we just presented the formalism of scenarios, one can easily perceive that the simple fact of gathering all the required resources (humans, equipment, etc.) for each evaluation session takes a tremendous amount of time. It becomes even more dramatic when you consider the fact that you will have to adjust some parameters from one session to another with the same scenario.
- it may be very *costly*: evaluating a system that relies on, and enables the collaboration of various devices to channel the collaboration of a variable number of users implies that for a relatively long period of time you will have to gather (and "use") all of those participants. Such requirements obviously imply important costs: the equipment has to be bought, leased or eventually borrowed; persons have to be "requisitioned" and you can have to dispatch the participants to properly recreate the conditions of the scenario you're interested in. In addition we have to keep in mind that for correctly evaluating a system you have to implement with a sufficient level of reliability to provide relevant results. This kind of implementation, besides the time that it takes, can force you to hire people to develop the required system.
- the great number of parameters of such situations can make them extremely hard to properly handle. Actually, the kind of scenario that has to be used for the evaluation induces a potential disturbance coming from external factors. Thus, it can lower the reliability and the value of evaluation results.
- scenario situations and events can be complicated, if not impossible, to put in place.

Once all these difficulties (which aren't exhaustive) are considered the evaluation of the PCSCW model can seem compromised. Despite this difficulty we absolutely needed to evaluate our model. Focusing on this perspective we decided to use an alternative to the evaluation in real conditions: the use of the simulator. This alternative gets rid of most of the issues we just presented. Indeed:

- the *time consumption* of a simulator is a microscopic fragment of the time needed for the real implementation of a scenario. It allows running simulations anytime (when you sleep, when you work, when you play poker with your friends), anywhere (given that your simulator doesn't need a supercomputer to work, you can use it on whatever computer you want, even if it's preferable to run your simulations on the same one to be able to coherently compare your results). Moreover, a simulator can make a part of the analysis you would have done manually, and then save a lot of time.
- the *cost* of the simulator and its use are also lower than the total cost of all the logistic required to actually lead the evaluation in real conditions. Thus, even if the development of the simulator induces an extra cost, it is rapidly absorbed by the savings done afterwards. Besides, the development of a simulator is often the source of new ideas and helps the development of the final application by forcing the designers and developers to earlier face some problems.

- as we're dealing with a simulator, the number and types of parameters are controllable (or at least manageable), it implies lesser disturbance and then an higher reliability over the evaluation results. Still, we know that by controlling the external factors or simply avoiding anything external to the mere simulation, we may confine the evaluation and miss unexpected behaviours. Given this consideration we have to stay aware of this potential inconvenient and drive our evaluation accordingly. However, by controlling inputs of the evaluation, we're also capable of introducing new features to the scenario and immediately get a feedback of their influence on the simulation and the system. This last point alone is, from our point of view, sufficient to tolerate the potential lack of "unexpectedness".
- last point but not least, a simulator is obviously the perfect tool to deal with situation extremely hard or dangerous to recreate (as it is done, no comparison intended, for nuclear weapons testing).

We have just seen the reasons that made us choose to use a simulator for our evaluation campaign. As we have already evoked the protocol of evaluation, it is now the appropriate time to describe how the evaluation will be supported by the simulator.

On the Figure 60 we have illustrated how the evaluation takes place with the simulator.

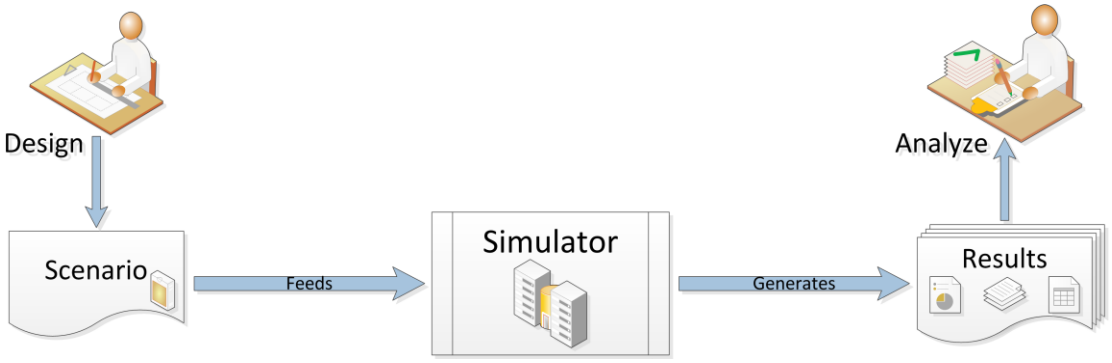

Figure 60 Evaluation supported by the simulator

At the left of the figure we've got the scenario designer which has in charge to create a scenario, as we have seen in the chapter dedicated to the simulator, scenarios are represented as ontologies. At the time we write this document we still have to create our scenario directly as ontologies with the Protégé platform, however it is totally conceivable to develop a specific tool allowing users unused to manipulate ontologies to create their own scenario via a more efficient interface. Once the scenario has been written, it is fed to the simulator which can run it. When the scenario come to an end or if the timeout is reached the simulator generates its results as a set of reports and traces of the simulation. These documents can then be directly analysed by an evaluator or stored to be analysed later. If the evaluation of the scenario requires some extra rounds, it is easily "re-runnable". Once all required simulations have been done and all of them have been analysed the evaluation of the scenario is completed.

9.2 Evaluation Cases

Designing an efficient evaluation protocol and providing a coherent configuration for the evaluation campaign are primordial requirements. However, even if these two steps are properly handled, the design of evaluation scenarios can still be problematic and can ruin your evaluation. In this perspective we tried to design scenarios with various settings, from simple cases with few possibilities to complex situations with "open" perspectives.

9.2.1 Scenario “Simple Connection Bridge”

This first scenario is based on a simple problem which can happen to any electronic devices user: reading a mail. Indeed, even if this action can seem completely trivial in everyday life, it is not always so simple.

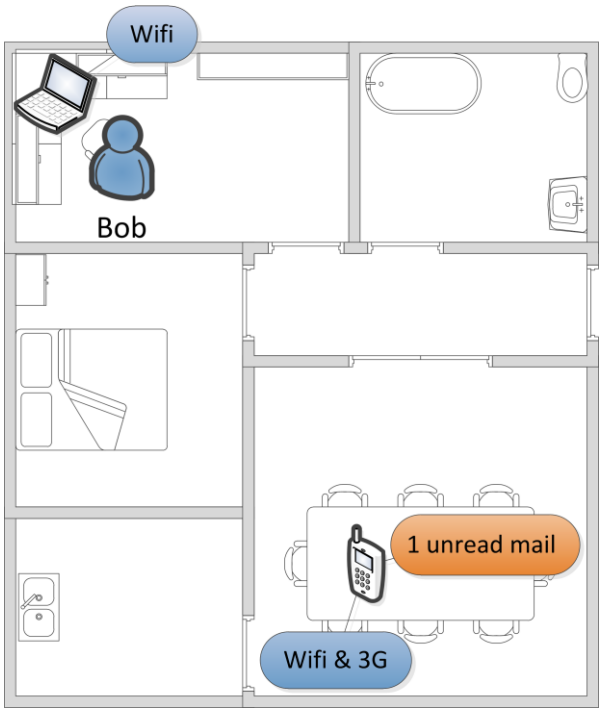

Figure 61 Simple Connection Bridge Scenario

Figure 61 gives a good insight of this first scenario. As illustrated we’ve got a human user *Bob* which is currently using a laptop in his office. In addition to the laptop we’ve got a second device: Bob’s cell-phone which is lying on the table of the dining room where Bob forgot it after lunch. This situation can be considered as the initial state of the scenario. To complete this description we have to precise the fact that Bob’s laptop doesn’t have an access to the Internet but has a Wi-Fi adapter. Bob’s cell-phone has a Wi-Fi adapter (currently off) and a 3G connection.

The rest of the scenario is composed of two simple events:

1. A new mail is fetched by bob’s cell-phone;
2. The scenario ends when bob reads his mail.

The first event of the scenario has for objective to activate the task “Read Mail” associated with the basic role of humans. And naturally the end of the scenario is marked by the completion of this simple task. Although one can think that this scenario is naïvely simple, we estimate that it represents a reliable start for the evaluation of the model. Indeed, and as we’ll see in the analysis of the results, our model can provide an interesting advantage for Bob and his collaboration with others.

9.2.2 Scenario “Multiple Choice Connection Bridge”

As we just detailed, our first scenario represents a relatively simple situation with a single human involved (even if the mail he receives has been sent by another one). For the second scenario we have taken one of the use cases we presented in Chapter 6 with some modifications. In this perspective we have written the scenario illustrated by Figure 62.

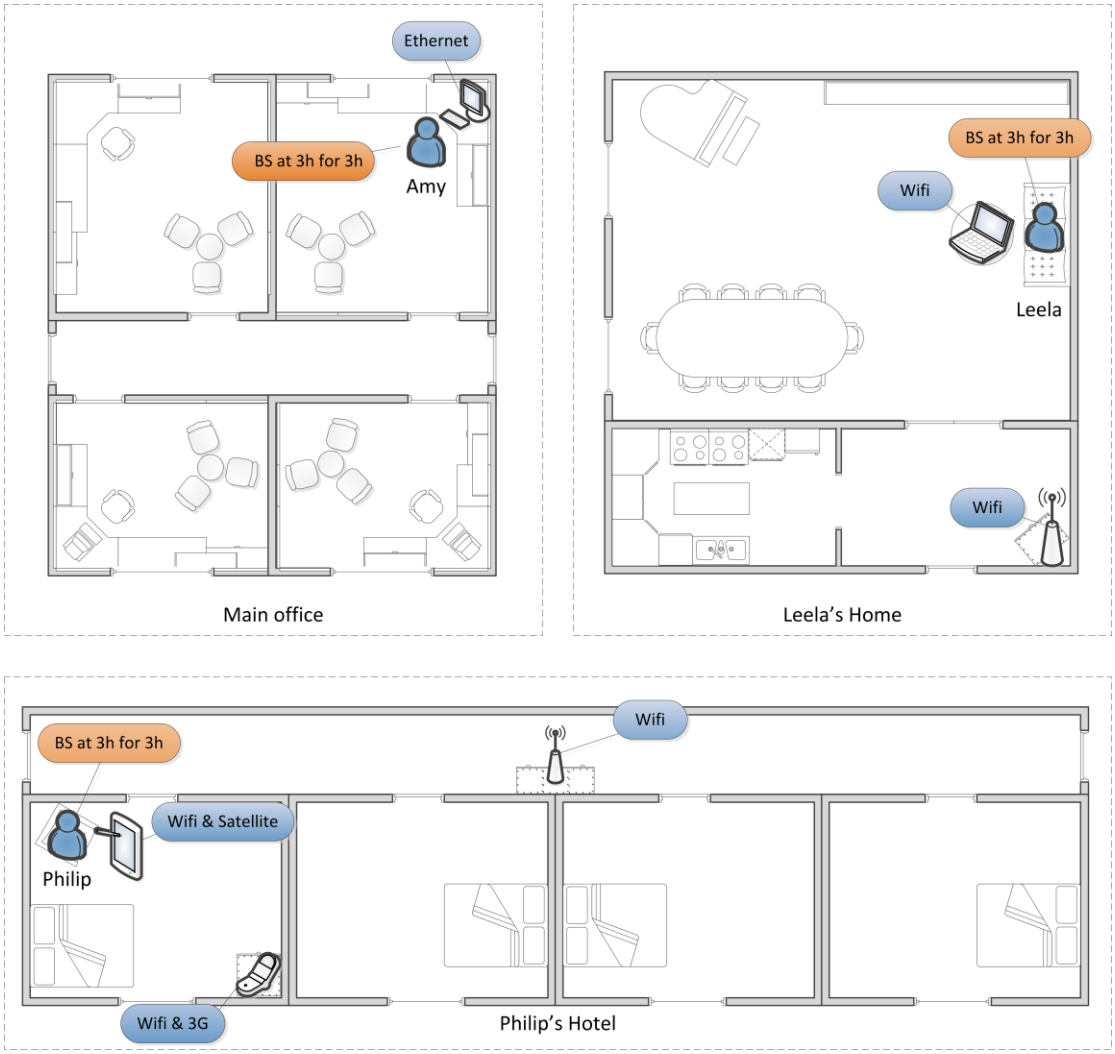

Figure 62 Multiple Choice Connection Bridge Scenario

As it is depicted, this scenario involves the collaboration of three persons in three different locations: Amy in the main office of her company, Leela in her home and Philip in a Hotel. The initial state of the scenario is the following:

- Amy, Leela and Philip are three team members of a marketing company. Their boss has just asked them to redesign a part of the advertising boards of a marketing campaign;
- As they're a little in the hurry for this task, they have to hold a brainstorming session as quickly as possible;
- They have agreed on an online meeting for the brainstorming session using a specific application running on a client-server model. The server part of the application is supported by one of the main servers of the company while the client part has been installed on each computer of the members;
- The brainstorming will be held at 15h and will last for three hours. At this time Amy will be at her desk in the main office of the company, using her usual workstation. Leela won't be able to go to the main office and will work from her home with her professional laptop. Philip is currently in mission in Kenya and will have to use his tablet-pc in his hotel room;
- The scenario starts some minutes before the time schedule of the brainstorming, Amy has her usual access to the Internet and has already started the application

on her computer; Leela's laptop is connected to Internet through the Wi-Fi access point of her home.

- Philip on his side isn't yet connected to the Internet as he's finalizing the preparation of the brainstorming. In addition to his tablet, Philip is equipped with a smartphone;
- The scenario will end with the brainstorming at 18h.

In addition to its initial state this scenario contains one event:

- After 1 hour and for 1 hour and a half the wifi connection is slowed to 0,5mbps due to multiple Internet accesses in the hotel;

This scenario is obviously more complex than the first one as it implies a scheduled collaboration. Indeed, in the first case we had a single person who had to read a mail *as soon as possible*. On the contrary, in this case the participants have to be ready at a given time and for a given duration. These two simple facts are in reality two more constraints for the collaboration. Thus we can summarize the real constraints to be fulfilled to reach the end of the scenario:

- Philip has to find a way to connect to the brainstorming application;
- Philip's connection has to be secured at least a minimum;
- This connection has to offer a sufficient bandwidth to correctly use the application;
- It also has to be reliable, the brainstorming experience needs all participants to be present during all the session;
- Thus it also needs to be sustainable for the three hours of the collaboration.

From all these consideration it becomes clear that the unfolding of such scenario isn't as simple as it looks. As we'll see later in the results of the evaluation this scenario can have several distinct ends.

9.2.3 Scenario "Architectural Firm Workflow"

For this third evaluation case we have designed a scenario implying the collaboration of people with different locations, equipment, roles and tasks. The Figure 63 illustrates the initial state of the scenario, with all its protagonists and equipment in their respective location.

9.2.3.1 Initial state

As it can be seen on the figure (Figure 63), the scenario involves four participants. *Hermes* and *Hubert* are two members of an Architectural firm located in Paris. They currently work for a contractor located in London who wants to build a Hotel in a new dynamic area. This work implies that they have to go several times a month to London to present the evolution of their work to the contractor.

Dwight is the Chief Architect of the architectural firm and has mainly in charge to review the design of the other architects.

Zapp is the contractor, in addition to his role of manager inside his company he also has in charge to receive architectural design and evaluate them.

The scenario takes place while Hermes and Hubert are going to London by train. As their journey will last for some hours they have decided to work in the train. For their work they use a heavy application that allows them to design and draw architectural plans collaboratively. To go to London, each of them has packed his laptop and has decided to work with it in the train.

The software they use, due to its collaborative aspect, allows them to work on the same project at the same time via a network connection. However, as the amount of data they manipulate can be huge, the connection between the clients of the application has to be as fast as possible.

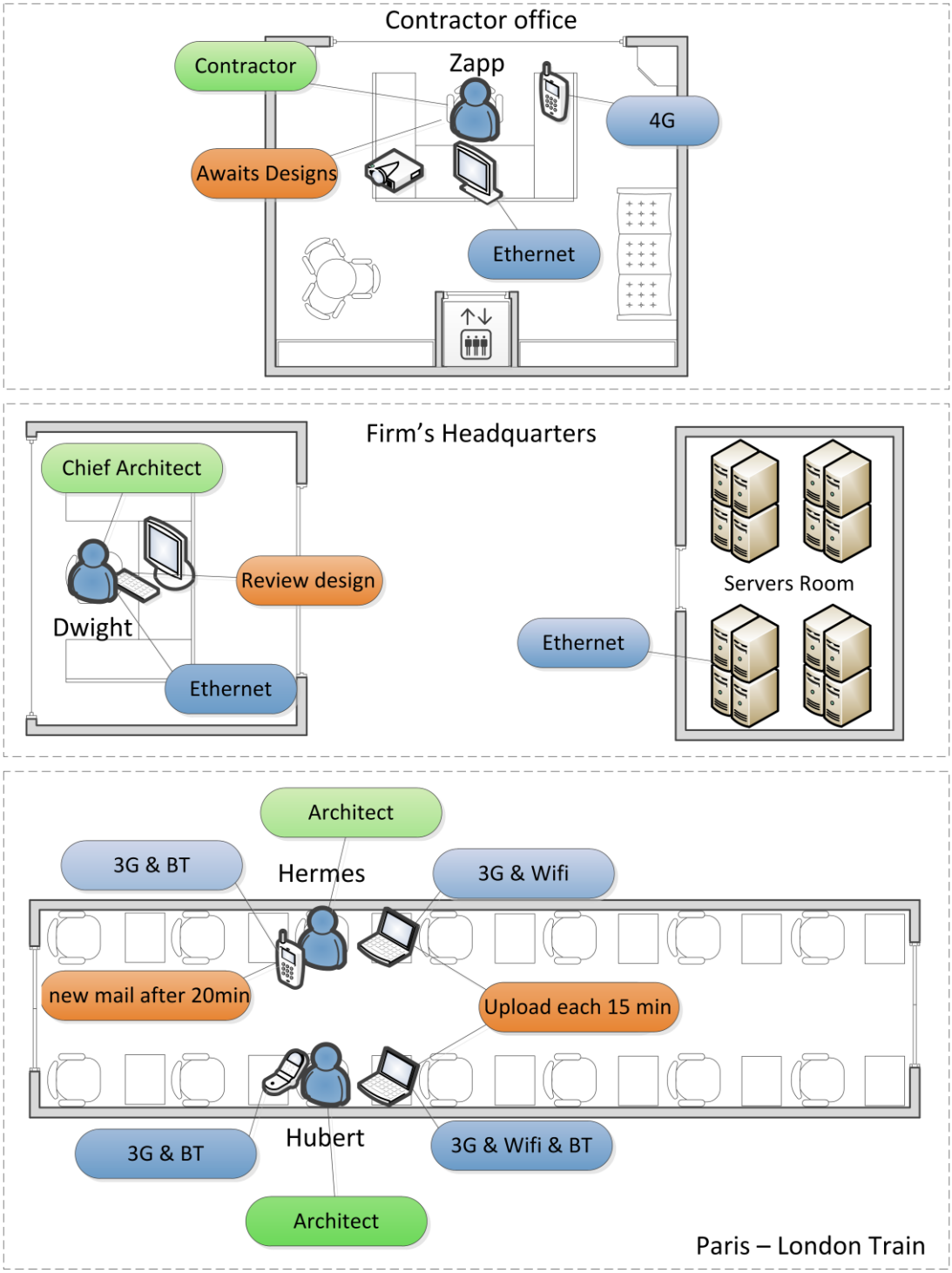

Figure 63 Architectural Firm Workflow

Let's summarize the situation at the beginning of the scenario:

- Hermes and Hubert are in a train that goes from Paris to London, their trip will last approximately two hours;
- They are both using their laptops connected to the Internet via their 3G adapters;

- They're working on the design of the Hotel they've in charge with their usual collaborative software;
- Dwight is working in the firm headquarters on his usual workstation;
- Zapp is working at his desk in his office.

9.2.3.2 Scenario Events

In addition to this initial state, the scenario includes a sequence of events that can alter the collaboration:

- After 20 minutes a mail containing information relative to the project on which they're working arrives on the phone of Hermes;
- Each 15 minutes the application saves the modifications of the project online. During the saving the project can't be modified;
- During the scenario, if Dwight opens the collaborative design tool and has to review a part of the work of Hermes and Hubert, he can start the review.

9.2.3.3 Tasks

The current work of Hermes and Hubert on the hotel's design implies a whole set of tasks to be completed sequentially. For instance, their meeting of today concern a part of the hotel they have completed four days ago. Once they estimate that a part of the design is completed they usually mark it with a specific tag and "seal" the design.

After that the chief architect (in this case Dwight) has in charge to review their work and approve or reject it. If the chief architect approves it, he has to send a copy of the approved design to the contractor to inform him of the evolution of their work. As they're close to have finished another part of the design they have planned to seal it before the end of their journey.

The scenario ends with the arrival of the train in London.

9.3 Evaluation Results

Now that we have presented the three scenarios we have used for our evaluation. And given the fact that we already described our evaluation protocol and configuration, the time has come to present the results of the evaluation we lead. For the presentation of these results we will proceed in three phases: the description of the unfolding of the scenario without the support of the PCSCW model, the description of the unfolding of the scenario with the support of the PCSCW model and finally a comparison of the scenario with and without our model and a discussion of these results.

9.3.1 Results of Scenario "Simple Connection Bridge"

The evaluation of the first scenario is relatively simple. Indeed as we depicted, it only contains a single event (the reception of a new mail). And a single task: the fact for Bob to read the mail.

9.3.1.1 Without the PCSCW model

If we consider this scenario without the PCSCW model, the unfolding of events can go like this:

1. A new mail from one of Bob's colleague arrives on the cell-phone;
2. As Bob is working on his laptop and isn't in the same room as his cell-phone he can't be aware of the new mail he just received;

3. After an undetermined period of time Bob or another member of his family check his cell-phone and see that he has received a new mail (if it's a member of his family, he will give him his cell-phone);
4. Once Bob has checked his cell-phone or has been warned by its family he can read the mail on his cell-phone.

9.3.1.2 With the PCSCW model

With the use of the PCSCW model (and assuming that it has been properly implemented and integrated), the scenario can unfold quite differently:

1. A new mail from one of Bob's colleague arrives on the cell-phone;
2. As Bob is working on his laptop and isn't in the same room as his cell-phone he can't be aware of the new mail he just received;
3. Bob's cell-phone explores its surrounding network by sending queries for context information to all reachable devices. In this case, the only reachable device is Bob's laptop, which replies by giving details about his context. Two information are important here: Bob is currently using his laptop and the laptop hasn't any Internet access;
4. Once the cell-phone has received Laptop's information he can reason over its context. A device collaboration is found and the cell-phone applies it;
5. He establishes a connection with the laptop with the use of the Wi-Fi network;
6. He also creates a connection bridge between its Wi-Fi and 3G connection, allowing the laptop to access to the Internet;
7. Then he sends to the laptop the information that a new mail has arrived on the specific mailbox;
8. The laptop can then automatically open the correct mail client of web browser on the mailbox URL and notify Bob (via a simple popup, or any other convenient way that Bob can have set up) that he has new mail to read the specified mailbox;
9. Bob can then read his mail.

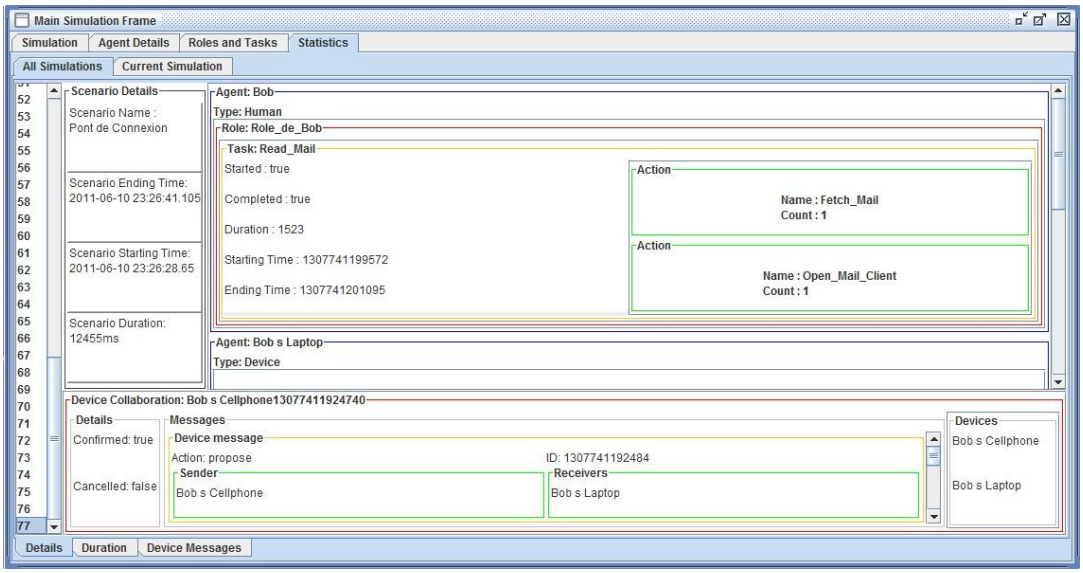

Figure 64 Simulation Report, General View

As we mentioned before, we developed our simulator in order to help us evaluate the PCSCW model. Let's have a look at how it helps us deal with the evaluation of this first

scenario. In addition to the obvious fact that the simulator supports the simulation of scenarios, it is of particular help when it comes to the analysis of what happened during the simulation. Indeed, as we already described in its dedicated part, the simulator provides a report of the simulation session. On Figure 64 we present the main window displaying all information about the past simulation: details about the scenario, list of actors with their roles, tasks and actions. In our case, these data are quite simple: the scenario lasted 12.455 seconds, it involved 3 actors: Bob (of human type), Bob's Cell-phone (device) and Bob's Laptop (device). As he's the only human, Bob is the sole to have an effective role, which is also quite simple as it only contains one task: "Read Mail" (which was started and completed). This task is itself decomposable into two actions: "Open Mail Client" and "Fetch Mail".

On the lower part of the window are presented the different device collaborations that took place during the simulation. Figure 65 shows the succession of the different messages that were sent by devices to set up the collaboration:

- The first message was sent by Bob's Cell-phone to Bob's Laptop: it contains a proposition for a collaboration between the two devices;
- In response to the proposition, the Laptop sends a message to the Cell-phone to accept the proposition of collaboration;
- After having received a response from the Laptop, Bob's Cell-phone has sent a final message to confirm and start the collaboration between the devices.

Figure 65 Devices Messages

This aspect of the simulator is particularly important as it helps us determine if the device collaborations are correctly managed. It is also critical in the evaluation of the model as it can tell us if the device collaboration rules used during the simulation produced the "expected" behaviours.

Even with this simple scenario it becomes clear that the ease brought by the simulator to test and evaluate the PCSCW Model and the designed Device Collaboration Rules is highly valuable for us and largely compensate the time that was invested in its development.

9.3.1.3 Comparison and Analysis

In the case of the unfolding without the PCSCW model, the time between the reception of the mail and the moment when Bob effectively reads it can vary. If we consider him "lucky", the mail can arrive when Bob was going to the kitchen to make himself a sandwich. He will then be able to read the mail as soon as it arrives. On the contrary if we consider Bob very "unlucky", he can simply forget to check his cell-phone until he goes to bed to setup the alarm for the morning.

This extreme variation of the reception of the mail can be avoided with the use our model. As we presented it, ever since the moment when the mail is fetched, the cell-phone tries to find a way to allow Bob to read his mail as soon as possible. Even if Bob has left his

laptop during the collaboration between his laptop and cell-phone (which can take place in a matter of seconds), there is still more chances for him to read his mail if he returns to his laptop than there was without the use of our model.

The different simulation we've done have shown that the collaboration between Bob's laptop and cell-phones can vary a little. For example, when the cell-phone asks the laptop for context information, the cell-phones can also sends information about its own context. Thus the laptop can simultaneously reason on the same set of context and potentially propose the collaboration before the cell-phone propose it. Apart for small differences like this one, the rest of the scenario rests the same.

If we compare the two versions of the scenario, the advantage offered by our model is relatively obvious: in most of the cases, Bob will be able to read his mail way before he could have done without. Even in this simple scenario we know that our model will not always be faster than the human action. Indeed if we consider the "lucky" case of unfolding, Bob can read his mail a little faster than he could have done if he hasn't moved from his laptop. However, even in this opportunistic case, if we consider the fact that there may be an attachment with the mail; Bob may still need his laptop to open it. In this consideration, giving Bob's laptop the opportunity to connect to the Internet and directly open the attachment would have noticeably facilitated Bob's task.

9.3.2 Results of Scenario "Multiple Choice Connection Bridge"

While the first scenario was an opportunity to show the advantage of our model in a simple case involving a two devices of a single user in a low constrained situation; this second scenario can give us an appreciation of the utility of our model in a synchronous collaboration situation involving three persons. Let's have a look at how this scenario can unfold.

9.3.2.1 Without the PCSCW model

As we already described, this scenario starts some before the actual start of the brainstorming session of Amy, Leela and Philip. Amy and Leela are already connected to the Internet and are ready to start the collaboration. Philip in his hotel isn't already connected to the Internet.

To establish the connection to the Internet Phillip (which isn't really a computing expert) has two possibilities:

- Using the satellite connection embedded on his tablet which has some advantages and drawbacks:
    - Advantages: high level of security, which only rely on tablet's energy and a relatively stable connection;
    - Drawbacks: slow speed (~0,8mbps), costly as each minute is charged and occasional disconnections happen.
- Using the hotel's Wi-Fi which also has advantages and drawbacks:
    - Advantages: good bandwidth (~2mbps), free of charge, low energy consuming;
    - Drawbacks: extremely poorly secured, the bandwidth may vary according to the number of other guests using the connection and each connection to the hot spot is severed each 15 minutes.

In these conditions and without the use of the PCSCW model the scenario can unfold like that:

1. Philip has taken his tablet-pc and is preparing of the brainstorming;
2. He can connect to the Internet with one of the two presented methods;

If he has chosen to use the Wi-Fi connection the scenario will unfold as:

3. Philip connects his laptop to the Wi-Fi of the hotel;
4. He his disconnected every 15 minutes, which greatly disturbs the brainstorming session as they can't keep their focus for a long period of time;
5. After 1 hour, the event takes place and the connection of Philip falls to 0,5mbps, which is hardly enough to maintain the brainstorming session and dramatically degrades the collaboration;
6. The last half hour of collaboration pass without any other incident;
7. At 18h the brainstorming sessions ends. Amy, Leela and Philip haven't been able to complete all their tasks and they had to schedule another session.

If Philip has chosen to rely on its satellite connection:

3. Philip connects his laptop to Internet through the satellite modem;
4. During the collaboration he experiences two disconnections, troubling the collaboration a little each time;
5. The slow bandwidth offered by the satellite connection also slows the collaboration as Philip is always the last to finish the download of brainstorming documents;
6. The brainstorming session ends at 18h. Almost all tasks have been completed, but Amy and Leela will have to meet again to put the finishing touches.

9.3.2.2 With the PCSCW model

Let's have a look at how it can have unfolded with the use of the PCSCW model:

1. As Philip hasn't yet establish a connection to the Internet few minutes before the start of his scheduled brainstorming session his tablet has to evaluate the best way to connect to the Internet and fulfil the constraints associated with the brainstorming session;
2. Philip's tablet then explores its surrounding networks for potential Internet accesses;
3. Its exploration finds the hotel Wi-Fi access point, Philip's smartphone and some other cell-phones and laptops, probably in other rooms of the hotel. Obviously the only two devices to which the tablet can "trustfully" connect are the Wi-Fi access point and the smartphone;
4. Once these information have been collected, the tablet can reason on its extended context and try to find the best way to connect to the Internet;
5. This reasoning gives him 3 potential connection schemes, the two possibilities already mentioned and a third one: as Philip's smartphone has a Wi-Fi adapter, it can connect with the tablet and create a connection bridge between its Wi-Fi and 3G connections. This third possibility has its own advantages and drawbacks:
    - Advantages: as the smartphone and the tablet are using an ad-hoc connection they can use a secured protocol for their exchanges, the average bandwidth delivered by the 3G connection and channelled to the tablet through the smartphone is relatively good (~1,6mbps), the connection is stable;

- Drawbacks: the power consumption of this configuration is high and at this rate the smartphone battery can only last for two and a half our;

6. Once these possibilities have been found, the tablet chooses the best one, according to the criticity of each constraint. The choice is made in favour of the third possibility (connection assisted by the smartphone);
7. Thus the tablet sends to the smartphone a collaboration proposition to the smartphone which accepts it;
8. The tablet establishes the Wi-Fi ad-hoc connection with the smartphone while this one creates the connection bridge between its Wi-Fi and 3G connections;
9. In addition the tablet notifies Philip that he should recharge the battery of his smartphone;
10. The brainstorming session normally starts at 15h;
11. There is no disconnection during the session and the bandwidth stays at an acceptable level, allowing Philip to work in good conditions;
12. At 17h, while there is only half an hour of battery left for Philip's smartphone, the tablet reminds to Philip that he should recharge his phone and inform him that if he don't do it, the Internet connection will be switched to a worse one;
13. Philip ask Amy and Leela for a small pause, and plug the AC adapter of his smartphone;
14. The brainstorming resume shortly after;
15. At 18h the session ends, all the points Amy, Leela and Philip had to work on have been treated.

9.3.2.3 Comparison and Analysis

Once again the comparison of the two unfoldings reveals important differences between them. If we refer to the initial set of constraints we had defined, the use of our model has allowed Philip to efficiently participate to the brainstorming session and complete the set of tasks he had to do with Amy and Leela. However, contrary to the simple case (without the use of the PCSCW model), the use of the smartphone to provide the tablet a better Internet connection involves an extra action for Philip: he has to plug an ac-adapter on his smartphone before the end of the collaboration. If we consider that Philip may simply ignore this constraint and don't recharge his smartphone the collaboration can still go on. Hence, when the smartphone's battery will be depleted, the tablet-pc can simply turn off its Wi-Fi and switch on his satellite connection or keep the Wi-Fi on and connect to the hotel's access point. In this case, even if the connection would be worse than with the smartphone help, it will only last for about half an hour. Amy, Leela and Philip may not be able to complete all their planned tasks, but most of them will have been treated.

In the simple case we have considered an unfavourable situation where the Wi-Fi access is slowed down for a long period of time. If the scenario had been a little different, for example if it was the 3G connection that has been slowed down, in the pcscw-supported case the tablet-pc could have acted in the same manner as if the Philip hasn't recharged his smartphone and could have switched the Internet access on another available connection.

With this simple scenario we have seen how the simple definition of constraints on resources allows our model to choose between multiple opportunities. Despite this quality, we can't expect our model to always find a perfect device collaboration that would ensure the good accomplishment of a task. If Philip hasn't had a smartphone with a 3G connection or simply if he had forgotten his AC adapter, he wouldn't have been able to properly complete the brainstorming session and in this case the only action that our system would have been

able to do is to automatically establish the Internet access with one of the two "simple" connections.

However, it is important to notice that in this scenario, the PCSCW system allowed the user to almost seamlessly participate to the brainstorming in better conditions that he would have without.

9.3.3 Results of Scenario "Architectural Firm Workflow"

We now reach the final step of our evaluation campaign as we'll present the results of our third and probably most interesting scenario. This scenario obviously offers more possibilities as it contains more humans, more devices, more roles, more tasks and more events.

9.3.3.1 Without the PCSCW model

This scenario hasn't any tasks to be completed before its end. It only ends when the train reaches London. In this perspective, the definition of constraints isn't as primordial as in the precedent case, even if it remains a major point of the model.

Due to the low level of constraints, this kind of scenario can have various unfoldings. Let's take an example of a potential one, not too pessimistic, neither too optimistic:

1. Hermes and Hubert have taken their places in the train;
2. They have started using their laptops to work on the part of architectural design they're currently on. They're both connected with their respective 3G cards;
3. After 20 minutes, Hubert receives an email on his cell-phone, after reading it on his cell-phone, he opens a mail client on his laptop to collect the mail and its attachment, he also forwards it to Hermes;
4. Each 15 minutes the application automatically saves all the modifications of the project which takes between one and four minutes;
5. Two minutes before the arrival of the train in London, Hermes and Hubert pack up their laptops and plan to finish their current design during the return journey;
6. The train reaches London and the scenario ends.

However we can consider a more optimistic variation of this scenario (starting back from step 4):

5. Some minutes before the arrival of the train in London, Hermes and Hubert finish the part of the design on which they were working;
6. Once they've quickly check their work, they mark it completed and seal this part;
7. Then Hermes sends a mail to Dwight (the chief architect) to inform him that they have finished their current work;
8. The train reaches London and the scenario ends.

9.3.3.2 With the PCSCW model

As we have showed this scenario can effectively work and unfold without the use of our model. However, we need to know if the support of the PCSCW model can help and facilitate the collaboration:

1. Hermes and Hubert have taken their places in the train;
2. They've switched on their laptop and Hubert has started to work on their project;

3. When Hermes on his turn starts the application and load the project, his laptop makes a routine context collection by querying all accessible devices for context information and search for a device collaboration by reasoning on the collected data;
4. Hermes' laptop finds out that, as Hubert is also working on the same project and that they're using a specific collaborative software requiring the highest possible network bandwidth, Hermes and Hubert' laptops should establish an ad hoc Wi-Fi connection to directly channel the communication of their architecture application;
5. Thus Hermes' laptop sends a collaboration proposition to Hubert's, which accepts it. The device collaboration takes place and fasten the communication between the two application clients;
6. Hermes can then efficiently collaborate with Hermes;
7. After 20 minutes Hubert receives an email on his cell-phone which searches for a device collaboration and finds an effective one: it quickly connects to Hubert' laptop and informs it that there is a new mail to read on a specific mail address;
8. The laptop then automatically opens the mail client dealing with this mail address and fetches the new mail;
9. As the mail client notifies Hubert of the new mail, he can read it directly on his laptop;
10. As the application saves the project each 15 minutes, both laptops looks for a device collaboration, they find the following possibility: as Hubert's laptop has a BlueTooth connection and his cell-phone also has a BlueTooth and a 3G connections it is possible for them to establish a BlueTooth connection and access the Internet through this connection by creating a connection bridge between the BT and 3G connections of the cell-phone. Thus it is possible for Hubert's laptop to use both its 3G and the cell-phone's 3G connections to fasten the upload of the project. Moreover, the two laptop, has they are connected through their Wi-Fi connections can efficiently balance the amount of data they upload each 15 minutes. In addition, Hubert's laptop can also communicate with Hermes' cell-phone over the BlueTooth connection. Thus it is possible for the two laptops to use both cell-phone's 3G connection in addition to their own. Finally, every 15 minutes the two laptops use the 4 available 3G connections to upload project's update;
11. About half an hour before the arrival in London, Hermes and Hubert finish their work on their current part of the architectural design. They mark it and seal their work;
12. At the moment they seal their work, the system evaluate the current context and figure out the fact that one of the tasks of Hermes and Hubert is completed. Given the fact that this task is a subtask of the general architectural design, the system knows that the next task to be performed is the review of the finished part by the chief Architect. Given this fact, the system can look for a way to contact the chief architect. By looking into Hubert's and Hermes' address books, the system find the mail address of Dwight which has been defined as chief architect. Then the system sends a mail to Dwight, informing him that the specific part of design has been sealed and can be reviewed, it also notifies Hubert and Hermes that a mail has been sent to Dwight;
13. Dwight's computer fetches the mail and detects that it concerns the review of a specific part of the architectural design that Dwight has to review. Considering

this state, the system automatically opens the designing application and the relative project. It also notifies Dwight that the design is ready to be reviewed;

14. Dwight then starts the review of the design;
15. After approximately 20 minutes Dwight has completed the review and approves the design proposed by Hermes and Hubert;
16. The system detects that Dwight has finished the review and automatically look for the mail address of the referring contractor;
17. Dwight computer finds the mail address of Zapp and automatically sends him a mail with an unalterable copy of the approved design;
18. Zapp receives the mail with the approved part of design at about the same time as Hubert and Hermes arrive in London;
19. The scenario ends, Hubert and Hermes have finished the task they were working on, Dwight has been able to review it and Zapp has already received the next approved design. He will be able to discuss on it with Hermes and Hubert when they will meet later today.

9.3.3.3 Comparison and Analysis

Once unfolded, we can see that there are interesting differences between these situations (with and without the support of the PCSCW model). Indeed in the first case, Hubert and Hermes have been hardly able to finish their current design (in a less optimistic case they may not be able to finish it), while with the support of our model they have been able to complete their work, Dwight has been able to review and approve their design and Zapp has received a copy of it.

Obviously the unfolding of the scenario with our model can be worse. For instance if the email address of Dwight couldn't be found, the system wouldn't have been able to notify him that the design is ready to be reviewed. Consequently Hermes or Hubert would have to send a mail to Dwight to inform him that they have finished their work.

From another perspective, the use of cell-phones' connections to speed up the upload of project's modifications represents another aspect of the support of the PCSCW model. Indeed, this device collaboration doesn't solve an unavoidable issue; it "only" improves the human collaboration by reducing the time of unavailability of the system.

Another interesting point in this scenario is the similarity we can point out with the first scenario we used to evaluate our model. This similarity comes from the fact that in both scenarios we've got a similar event which is handled in a slightly different way. Indeed, when Hubert receives a new mail on his cell-phone he is in a comparable case as was Bob in the first scenario. However the main difference between these two scenarios is the fact that Hubert laptop has it own Internet access. This little difference in the situation induce a little difference in the collaboration of devices: while in the first case the two devices were forced to collaborate for an extended period of time to channel the Internet connection to Bob's laptop to allow him to read his mail, for this last scenario the cell-phone only has to quickly notify Hubert's laptop that a new mail needs to be read and then let it handle the rest of the collaboration.

Finally, this scenario represents a good example of the use of the description of tasks and roles in our model. This use is illustrated by the workflow taking place automatically between Hubert, Hermes, Dwight and Zapp. Indeed, if we hadn't described the architectural designing task with the different steps it contains and the roles that are associated with each of these tasks, the system wouldn't have been able to know what to do when Hubert and Hermes finished their work or when Dwight approved the design. Thus this scenario shows us that in

addition to the more basic collaboration patterns, the PCSCW model is also a natural support for workflow mechanisms.

9.4 Quantitative Indicators

The evaluation we led on the PCSCW model has been oriented to mainly provide a qualitative evaluation of our work. As we already discussed, the development and the use of a simulator was the best way to evaluate our model as it offers many advantages over an evaluation in real conditions. However, we know that to complete the evaluation of our model it would be interesting to evaluate it in real condition with a fully operational implementation of our model. For example it could give us more quantitative indicators of our model. In the perspective to be able to compare the simulated and real evaluations and in order to illustrate the use of our previously presented indicators we have used the simulator to provide us the quantitative indicators presented on Tab 13:

Indicator	Scenario 1	Scenario 2	Scenario 3
Number of Humans	1	3	4
Number of Devices	2	4	6
Number of Tasks	1	1	5
Number of Actions	2	6	8
Number of assisted Actions	1	3	6
Number of automated Actions	1	1	2
Number of assisted Tasks	1	1	3
Number of automated Tasks	0	0	2
Number of device collaborations	1	1	5

Tab 13 Evaluation Indicators

As we can see on this table the quantitative indicators we are able to provide with the use of the simulator are a limited to the numbers of actions and tasks that have been facilitated or automatically performed with the use of the PCSCW model. As expected and consistently with the evaluation results the first and second scenarios, even if they are quite different, have similar statistics. Indeed, as these scenarios are mainly based on the execution off a single task, the potential improvements are relatively limited. Even though, we can notice that almost each task of the collaboration can be facilitated by the use of our model.

These quantitative indicators have an interesting aspect: they allow us to express the relative efficiency of the PCSCW model according to the number of humans, devices, tasks and actions. Then, they can help us compare the evolution of a scenario according to simulation settings or even compare different scenarios.

9.5 Evaluation Synthesis

The evaluation of complex systems is always a hard phase of a development. Indeed, when you consider a system with such a wide range of possibilities as the computer supported collaborative work, you can await to have severe difficulties to properly evaluate it. In addition to the collaborative work aspect, the PCSCW model brings the pervasive one. Such addition obviously implies a higher complexity in the evaluation of the system. In order to help this evaluation we have defined some guidelines to define indicators according to five aspects: *Simplicity*, *Efficiency*, *Reliability*, *Security* and *Quality*. The evaluation itself has given us interesting results and insights on the PCSCW model effectiveness. Indeed we have been able to show that our model can assist most of the tasks of the collaboration. The use of a simulation tool, even if it has its own drawbacks, allowed us to correctly evaluate our model

and adjust some our device collaboration rules in a small fraction of the time and cost an evaluation in real conditions would have taken us.

Part 3: Discussion

Chapter 10 Discussion

We have looked through the Computer Supported Collaborative Work by successively focusing on most of its different aspects. As we have seen CSCW is not new, collaboration has been an active domain almost since the beginning of modern computing. The pervasive computing paradigm on its side is still a relatively young domain that isn't yet integrated to the everyday life. From our point of view the pervasive computing paradigm is an unavoidable evolution of modern computing which will bring us to new ways of behaving and new ways of collaborations. When we started our work around the "combination" of the two concepts that are CSCW and Pervasive Computing we didn't know exactly what we would achieve. Our only thought was that gathering and combining these two aspects had to be possible and would reveal unexpected horizons.

From a collaborative modelling perspective our work does not bring a revolution, indeed most of our contribution is the way we combine and use sub-models. As we evoked in the second chapter our first approach to the computer supported collaborative work modelling was the consideration of different role models. This kind of model deals with the roles played by users in defining and describing their permissions, duties and interdictions.

Our thought is that for correctly modelling the collaboration between users you have to know how they have to interact with each other, how they are organized as a team, as a group, as a an enterprise, as partners, etc. The modelling of these interactions requires the definition and the description of roles of users in the collaboration. However, even if roles are or prime importance they're not the first element to define and describe. Indeed, the collaboration of users isn't roles-centred; humans firstly collaborate to reach a common (or shared) objective. This objective can be a direct one (collaboratively writing an article) or an indirect one (collaboratively designing a client/server software interface in order to be able to develop the client side). Given this consideration we can deduce that the first elements to define are tasks. The representation of tasks allows the precise description of humans' objectives. As we presented, task are defined as meaningful realizations of users (such as designing floor plans of a building), composed of a set of mixed actions and/or subtasks. Actions are "meaningless" realization of users (such as opening a web browser), while subtasks have to be considered as any other tasks, except for the fact that they're taking part in a greater realisation (a "supertask"). Relying on this simple task model we can depict any task to be performed by humans. Once this base is laid we can start adding the role model. Thus, if we roughly consider it the role model we use builds roles by forming group of tasks. A role is then defined by tasks it has to perform, it can perform and tasks it cannot perform. Even if this principle seems too simple we believe that such description is sufficient to allow the correct representation of collaboration. Indeed, if two persons share a task (for example if they each have one of the sub-tasks of the main one to realize), it implies that they will have to collaborate at some point. This quite simple representation can obviously be augmented with other concepts. For instance we can consider the addition of "group roles", gathering a set of roles into a greater one to represent which tasks a group of persons have to do. This kind of concept can also be useful to define roles shared by several users and implying the fact that tasks can be performed by any person, not by a precise one.

From these considerations it becomes clear that the base of the PCSCW isn't closed. On the contrary we believe that an efficient model has to be able to be freely and endlessly extended and enhanced. This specific part of our work takes another dimension if we relate it to the resource model. As we have presented, in addition to the role and task models, the PCSCW relies on the description of resources. This description is used at two different levels of our model. In a first time the description takes place in the representation of action. Indeed, we associate with each action of the task model a set of resources which are required to allow the accomplishment of the action. In a second time the resource description is naturally used to represent the current context of users. Given this uses, we have no wise choice but to

assume that the representation of resources is completely open. It would have been useless to define such kind of model and assume that there is no possible extension.

Despite the fact that our model is "open", it doesn't mean that extensions can be made thoughtlessly. Indeed we know that letting an open model evolve without fixing boundaries from the beginning may bring unexpected and interesting evolutions, but it will also more surely bring chaos and decoherence to the base of the model. In this perspective it is critical for anyone wanting to extend our model to preserve the general coherence of the model in building his new contribution in relying on the already existing base of the model. This constraint can seem restrictive; however it is an "absolute" necessity, at least if anyone wants to benefits from the already developed model and collaboration rules. Nevertheless, even the base of the sub-models may need to evolve (for instance if the resource model requires being able to represent new kinds of devices, new features, unrepresented social context elements, etc.). In this case it would require carefully inspecting device collaboration rules in order to evaluate if some of them have to be partially rewritten or if new rules need to be added. This requirement may seem inelegant or complex, but in fact it is an opportunity for developers to re-evaluate rules they have written and examine them in the light of the new model's elements. This kind of re-evaluation can help finding new device collaboration rules and then new ways for devices to behave.

When resources associated with actions have been defined and described, the last step to complete the representation of the context is to define constraints on these resources. As we have quickly evoked in the dedicated section, the definition of these constraints can be of critical importance in the processing of context information. Indeed, they can help devices decide of the best way to collaborate and how to behave to fulfil collaboration expectations and users' needs. Indeed, these constraints can be the expression of different aspects of the collaboration. For instance the collaboration between two persons can be subject to confidentiality or secret, the behaviour of devices then has to take this fact into account. To do it we can for example define a constraint on the connection they will use and require a high level of security. This confidentiality could also have been expressed by the encryption algorithm of documents they exchange.

The definition of these constraints is probably one of the most critical but also one of the most complex tasks associated with the use of the PCSCW model. Indeed, if you want to describe each constraint on each resource of each actions of each task associated with each role, it can become unmanageable, time consuming, costly and horribly repetitive. Despite this actual risk for people desiring "too much description", the PCSCW model doesn't *absolutely* require the definition of constraints on resources. Actually, the use of this element of the model is truly effective in highly restricted situations (for instance when the collaboration has to deal with confidentiality or very limited resources). We are aware of the fact that constraints are also valuable in the case where we've got multiple device collaborations matching the current context of the user. Still we think that defining every constraint of a collaboration situation is nearly impossible and can bring more restrictions than opportunities. Given this perspective the PCSCW model has been designed to be able to work without the expression of resources constraints. If we consider such case (which has been illustrated through the first evaluation scenario) the system will simply reason on its context and find matching behaviours. If only one matches, then it can be applied (or proposed). If several behaviours are found, given the fact that they are all supposed to be effective, the system can simply pick one of them and apply it. Thus, the resulting collaboration may be less optimized than if constraints had been expressed before the start of the collaboration, but it still will be more effective than without the use of the model. In addition, the definition of hard constraints on a relatively low restrictive situation can prevent most "exotic" (but potentially the most efficient) devices collaborations to be applied. For all these reasons the definition of resources constraints has to be wisely considered before their effective design. In order to tackle the difficulty of defining constraints, we think that an opportunistic angle of reflexion for the facilitation of the use of our model would be to

evaluate the possibility to describe constraints templates that could be used to simplify the qualification and quantification of constraints. Thus, the addition of constraints on resources can be seen as a non-mandatory complement to the rest of the model which allows a better adaptation to the current context by providing useful data allowing the system to choose between multiple behaviours.

Even if the representation of the context of users is an essential requirement for the correct adaptation to context and then to the correct functioning of pervasive computing solutions, it isn't the only feature that composed the heart of our work. The Device Collaboration Rules and their functioning are the second elements of our contribution. As we presented them, these rules are designed to provide to the system an "optimal" behaviour according to the current state of the collaboration. As they have been described, the main principle behind device collaboration rules is to compare current context resources against resources required by actions and to provide the most adapted behaviour for devices. The obvious main issue is the difficulty to design small, effective and reusable rules. Our deep though in regard of this difficulty is that small, really efficient and reusable rules aren't so complex. Indeed, if we consider the collaboration patterns usually taking place and the different way for devices to interact, it is relatively clear that these interactions can be analysed and redundant patterns can be extracted to provide building blocks for device collaboration rules. Once again we argue that this relative complexity is in fact an opportunity for collaborative systems designers to have another look at their domain and help its formalisation. Even though, as we designed some rules for our researches and our evaluation campaign we are very aware of the inherent difficulty to build such rules and it has to be kept in mind that for anyone wishing to use our model it will doubtlessly be a challenging issue.

The development of the PCSCW simulator, as a major step of our work, was of prime importance for us. As we already explained, we choose to develop a simulator instead of directly developing a "real" system in order to avoid multiple complex issues that would have dramatically extended the time required to actually implement the PCSCW model and even more the time needed for the evaluation. Besides, the simulator offers interesting features that can't be realized with a real system for a real conditions evaluation. The architecture we proposed and implemented for this simulator isn't perfect, and there would have been many other ways to implement our model and the simulation part itself. However, as it fits our basic expectations this simulator is a useful tool that already allowed us to modify some of the device collaboration rules we designed, as we figured out through the simulations that there were some lacks or misconceptions of these rules. In this perspective the simulator is a powerful tool that should be used to help the design of new device collaboration rules and allows designers to immediately test and evaluate these new rules by running sets of evaluation cases and analyse the different gains and losses. Thus, the simulator we developed can be considered as an evaluation tool and as a laboratory that helps developing new collaboration rules. Given this dual nature, the simulator has become an unavoidable resource that has helped us and will continue to help us in our future works.

The evaluation campaign we lead on our model had for goal to determine if the model we propose is effective and can improve, or at least facilitate, the collaboration of humans. In this perspective, the different evaluation scenarios that we considered have shown that the use of our model can simplify (at least in some cases) the collaboration of humans. We know that basing our evaluation on the use of a simulator has its own inherent drawbacks, and that it is unable to mimic the real behaviour of humans or the reactions they could have in the evaluation situation. However we think this evaluation is coherent, consistent and sufficient to establish that the PCSCW model can effectively work. As we have said in the evaluation chapter, even if we can tell if the model is effective or not, we can't honestly quantify the gain it may bring. Indeed, the simulation cannot deal with human uncertainty and free will. Given this consideration, it is clear that a real world evaluation would give interesting insight and indicators on the quantifiable gains our model can bring. Even though, the real world evaluation cannot replace the simulations that we propose. As we already argued, the use of

simulations allows us to easily run and rerun scenarios in the same time it would have taken us to prepare a fraction of the logistics of a real world evaluation for a single scenario.

In the general goal of integrating the Pervasive Computing paradigm with the Computer Supported Collaborative Work we have proposed a method that allows developers of new systems to correctly plan and organize their evaluation. By the description of a building method for evaluation strategies we propose an original way to deal with the evaluation of pervasive collaboration systems. As we rely on the definition of a taxonomy of evaluation methods our work merely intend to organize and reuse the work done by other researchers but also to make good use of the evaluations feedbacks provided by developers. We clearly know that this building method is a long shot and that it will take time to properly use. In addition we're also aware that applying it roughly can be too costly for most of developers. However we assume that this method doesn't force developers to use the complete strategy it proposes. Indeed, developers can simply choose to use some part of them according to their actual needs and resources. Still we think that the main principle of our method represents an interesting and valuable advance toward a better management of pervasive systems evaluation.

Another point we haven't addressed yet as it stands out of our focus is the problem of data collecting and interpreting. As we have presented, our model is based on the representation of users' context. This representation naturally has to be feed by data from context sensors and monitors. Without access to such data it is obviously impossible for any implementation of the PCSCW model to correctly work and adapt devices behaviours. Moreover we have noticed that the representation of the context can be highly descriptive and then require intense efforts and massive data streams to effectively work. However we know that context sensors are more and more effective and lightweight. This should solve at least a part of this problematic. In the same perspective we think that some work can be done on ontologies; as we'll detail in our future works we've started to work on some interesting tracks toward the reduction and scalability of ontologies. In this early work we've figured out that the representation of context information inside ontologies can be inefficient when context information are sparse. Indeed if the current situation of users' is simple the complete description of the context can be unsuitable. This fact can be even more noticeable when users' devices have limited resources (such as old cell-phones). Thus, one of the tracks we've got to reduce the size of ontologies is to represent information dependency between context's concepts. By doing so we think that we'll be able to avoid the representation of unrequired concepts and then to significantly reduce the size of ontologies.

Now, if we look at our work from the strict Pervasive Computing point of view, it is also clear that we do not address it in all its complexity and possibilities. Given our approach of the subject, we considered that by focusing on the generic improvement of the continuity of service, we needed to "mask" those services (such as location-based services …) and thus to leave them aside from our research. As a matter of fact we think that this exclusion is a benefit as it may allow us in future works to consider these services with the new perspective of our model, without having mixed them from the start at the high risk of a biased design.

The PCSCW model is our approach of dealing with the combination of pervasive computing and computer supported collaborative work. Obviously it isn't the only valid approach to deal with these two concepts, and there are doubtlessly numerous other ways to combine them. The model we propose has its own drawbacks; it can be overly descriptive, hard to handle and manage for developers. However, the benefits it brings easily overcome its potential drawbacks. Indeed, even if collaboration designers and administrators can have a hard time at the beginning of their use of the system, it should never deteriorate users' experience. Actually, as the PCSCW model is based on the principle of automatically adapting devices' behaviour to users' context, it obviously tries to limit actions of users to unavoidable ones. If we look a little deeper, we can find an underlying objective of our model: optimizing the use of resources for the user. To consider this perspective we can simply make the observation that most of the time humans are only using a small part of their resources.

Now, if we add the pervasive aspect, humans have to deal with multiple devices among which he may not know the functioning or the use. We can also mention the fact that with the representation of context information and notably collaboration information, and by providing a simple way to deduce device behaviours, the PCSCW model can be considered as a unifying model allowing the representation and the use of specific domains information. Given this aspect, the PCSCW model is a precious bridge which can serve as way to mediate information between users, their group and their electronic and smart environment. Finally, by representing all these context information and trying to find the best way for devices to behave we simply try to find the best way for humans to efficiently use all these smart devices and then to rationalize and optimize the use of available resources.

Conclusions and Future works

The PCSCW model has been designed to naturally integrate the pervasive computing paradigm within the computer supported collaborative work. In this initial perspective we gave the PCSCW model an ontological model allowing the representation of users' context, containing all information relative to their physical, temporal, computational, social, collaborative environment. In addition we gave to this model the definition of devices collaboration rules that are supposed to deal with current users' context and find the best match to perform their assigned tasks. To help its evaluation and the development of the model we designed and developed a simulator allowing us to play collaboration scenario and see how smart devices could collaborate through the use of the PCSCW model. Thus, this simulator has allowed us to conduct the evaluation of our model. This evaluation has been mainly a qualitative one. Indeed, due to the fact that a simulator cannot completely recreate a real situation and surely can't simulate humans with all their complexity, it wasn't relevant to found our evaluation on quantitative aspects. Even though, the qualitative aspects of the evaluation have been sufficient to highlight and prove the qualities of our model. Thus, the very nature of our model clearly appears in our evaluation cases which have shown that the use of our model can effectively simplify the work of humans by making devices automatically collaborate and then adapt their behaviours. One can argue that our model implies a high level of description which requires too much work before the start of the collaboration. On the contrary we think that such description is unavoidable in collaborative work situations and that it is already done, at least informally, in most of these situations. Going down this road we can argue that describing and formalizing the collaboration in every day's situations and feeding the PCSCW model with this description can be rapidly valuable for collaborators. Indeed, in order to improve the different sub-models and the device collaboration rules of our model, it is necessary to collect and analyse feedbacks from its use. Thus, the analysis of feedbacks can quickly help the evolution of devices collaboration rules and then the improvement of the collaboration. Finally, given its nature, its necessity to evolve and its native extensibility, the PCSCW model represents an efficient and perennial approach to support the pervasive computing era within the computer supported collaborative work.

Future Works and Perspectives

This research has pioneered a new approach to modelling and evaluating CSCW systems in pervasive environments. At the same time, it raises a number of research issues worth exploring in the future.

I. Short Term Perspectives

One of the first perspectives we've got for our model is to define new rules helping the system to choose between multiple device collaboration. Indeed, as we mentioned previously, the description of constraints on resources can be complex and highly time consuming. In order to be able to avoid most of this description and still correctly choose a relevant device collaboration, we need to define how it can be done. This kind of mechanism can, for instance, be based on the number of devices interactions involved by each rule. According to the current context of the collaboration or the current state of devices (battery level, connectivity…) it can be interesting for the system to induce more or less complex interactions. Thus, by analysing such behaviours it is possible to deduce some general rules specifically designed to help the device collaboration selection. To take a simple example: most of the time it is preferable for a device to have access to the Internet, as long as it doesn't compromise its battery life to critical levels.

When we started this work we had in mind to couple the computer supported collaborative work with the pervasive computing paradigm. As our researches progressed, we were dragged in the deep technological layers of modern computing and came out with a model that suited our needs. An unforeseen consequence is that if we take a step back and put this model in perspective with some other domains or situations, we can immediately see that it can span and be useful in other aspects than the one of the collaboration. Indeed, the PCSCW model could be used to support the continuity and quality of services for more classical mobile applications on a similar way it does for collaborative one. One of our possible futures would then be to investigate this extension to any mobile application and see if some more features are necessary to completely deal with them.

Another perspective for us is to lead another campaign of evaluation with the help of real users which could bring us their own evaluation scenarios. This kind of evaluation would give us interesting insight both on the ability of our system to adapt to new situation and on the expectations of users for their collaboration.

A valuable and relatively simple addition to the simulator would be to integrate a scenario edition aspect. This new feature would allow us to directly edit simulations scenarios inside the application and then to simplify the design of these scenarios by running simulations and adjust their parameters and composition.

Until now the description of device collaboration rules has been made relatively "freely". However we think that a valuable perspective for these rules would be to find a way to organize them. Such organization can be brought by different means and aspects of our model. Our actual thought is that this organization could be done by categorizing these rules according to the resources they manipulate (for instance we could make a category with rules dealing with Internet connection issues). We also think that the development of rules would be simplified by the creation of rule blocks that developers could assemble to create new rules. Thus, the addition of a new rule brick would immediately extend the rule construction possibilities. In the same perspective, but on a longer range designing and categorizing device collaboration bricks should allow us to design a system allowing the automatic building of device collaboration rules.

II. Ontology Scalability

The size of ontologies in the perspective of representing all information of users' context in a pervasive environment can dramatically grow and become hardly manageable by small devices. In order to solve this issue we need to find a way to reduce the weight of context representation. This reduction should ensure the ability to manipulate context information and reason over these information even on small devices with limited resources.

Even though we need to reduce the size of our ontology it is not possible to simply "forget" a part of the context. Indeed, you can't know when you will have to use this or this information. Thus, keeping in mind the work that has been done by **[Elmasri et al, 2007]**, we try to organize pervasive computing context in multiple levels. Doing so, we rapidly can come to the conclusion that a great amount of information is redundant.

Figure 66 Information Dependency

Our main idea to reduce the size of such ontology is to represent existence dependencies between classes, that is to say: which class requires an instance for the existence

of an instance of another class. Again, let's illustrate this principle by a simple example on Figure 66.

On the precedent figure we have represented some of the implication induced by the existence of the action "Browsing WebPage". Indeed, the simple fact of knowing that a user is browsing a webpage has a set of direct and indirect involvements: firstly if the user is browsing a webpage it implies that he is using a web browser; this web browser has to be supported by an active web connection which itself requires a network adapter embedded in a device. These cascading requirements are a simple but eloquent example of information redundancy. Indeed, if you only need the "web browsing" action you will have created the representation of, at least, three useless resources.

To help defining what instances can be "obscured" we propose to organize resources representation in layers. As depicted by Figure 67 on the top layer we've got containers that are absolutely required for the existence of sub layers elements. On the second layer we've got the main components embedded in a container. These components provide functionalities represented on the third layer. For each functionality we've got a set of associated data that forms the fourth layer. In a convenient way we can have several "levels" of components or functionalities according to the accuracy of the representation.

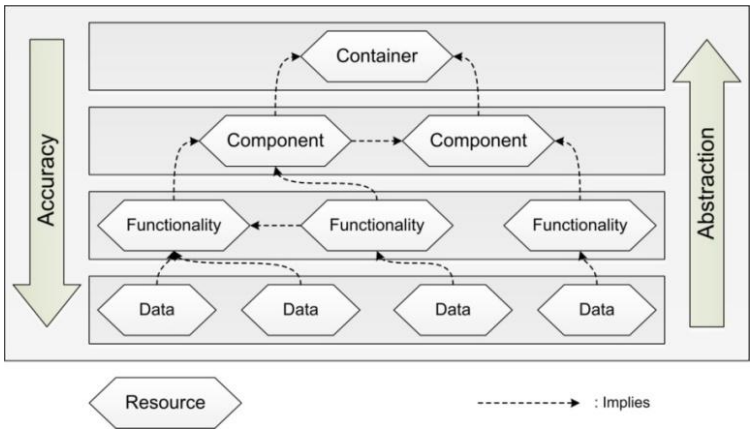

Figure 67 Layers and Dependencies

If we apply this formalism to the PCSCW model, we can quickly figure out that all our sub-models are organized according the previously evoked layers that represent successive levels of context abstraction. For the resource description part we can identify four main levels:

1. The root level, corresponding to the device itself for the hardware part and to the operating system for the software part;
2. The second level corresponds to main components: applications and services for the software aspect and peripherals (integrated or not) for the hardware part;
3. The third level depicts functionalities offered by each components of the precedent level. For instance a web browser can offer various functionalities to browse the web, history saving, favourite pages management, forum monitoring, etc.; while a network adapter can support different standards of Wi-Fi;
4. The last level of the resource model is the representation of data. It obviously relies on the third level as it is the expression of the current value of functionalities. For instance a network adapter can be connected or not to the network.

The task sub-model is also organized in several layers:

1. The top-level layer represents users taking part in the collaboration;
2. The second layer is a representation of their associated roles;
3. The third layer is composed of the tasks contained by each role;

4. The fourth layer represents actions required by tasks;
5. The fifth layer is probably the most interesting as it depicts the resources required by the actions to be realized.

Thus if we refer to the main principle behind the PCSCW model, devices have to decide how they have to cooperate to properly channel the collaboration between users. This is done by a comparison between resources required to perform an action and resources available in the current state of the environment. Thus optimally we have to map each required resource to an available one. In the perspective of reducing the size of the context ontology the layers we have defined can help us automatically infer information according to the deepest level of abstraction.

As we evoked previously, knowing that a user is performing the action "browsing the web" implies the existence of several resources (web browser, web connection, network adapter...). Then in an optimistic view we could say that the action surpasses the description of resources and that we can avoid to represent all implicit resources. We think this assumption is only partially right, indeed if we keep on considering our web browsing example it can be problematic if we completely remove the network adapter representation from the current ontology. Indeed even if the "web browsing action" can be sufficient for a part of the reasoning made on context knowledge, it can be insufficient if you have to deal with more specific mechanisms requiring the precise value of connection setting and functionalities. For instance if current user's task involves an action requiring an high speed bandwidth (such as video conferencing), knowing that he's already surfing the Internet doesn't bring sufficient information to determine if the network connection suits the requirements.

To solve this problem we propose a simple mechanism based on the analysis of reasoning rules. It can be summarized as following: "If a resource A implies the existence of a resource B and no reasoning rule contain a condition involving a data of B uncharacterized by A; then all references to B in reasoning rules can be replaced by A". This mechanism ensures that the resource B can be replaced by the resource without losing reasoning capabilities.

The last step on our road concerns the evolution of our simulator. We propose to integrate real devices into the simulation and allow them to interact not only with other real devices but also with simulated ones. We propose an architecture to support both real and simulated devices. This specific approach provides us interesting aspects in term of evaluation and promising perspectives for the development of pervasive computing behaviours.

III. Hybrid Architecture

The evolution we propose for our simulator is to integrate real devices into the simulation and allow them to interact not only with other real devices but also with simulated ones. Even if it seems interesting and promising this approach requires some modifications of the existing simulator's and agents' design to be deployable. In this perspective we'll now describe our view of a hybrid architecture for a multi-agent pervasive computing supported collaborative work simulator allowing real and simulated agents to interact and play a collaboration scenario.

Figure 68 depicts the architecture of a hybrid agent running deployed PCSCW services and interacting with JADE simulation agents. The architecture itself is organized in three main layers:

- *PCSCW Core*: this layer is the real core of the agent; it is composed of the following components: *Knowledge Manager*, *Knowledge Interface* and Behaviour Manager. As these components correspond to parts of the architecture we already detailed we won't give any more details about them.
- *PCSCW Deployment Core*: this layer of the architecture contains base elements to deploy and use the PCSCW model on a real device. Indeed, even if the PCSCW Core is still

required to manipulate agent's knowledge, it is not sufficient for a real deployment. Then, to be able to deploy it we need the following components:

- *Context Broker*: this module has in charge to handle the collection of context data which are not directly available to the agent. For instance, it provides common services to exchange context information with other agents;
- *Introspection Engine*: this component relies on information miners and collectors to give to the agent information about itself.

- *PCSCW Simulation Core*: this layer has the buildings blocks to enable the creation of a simulated agent. The reason why this part of the architecture is on the same level as the deployment core is that it shares a common goal: dealing with context information sources and feeding the *Knowledge Manager*. In this perspective, it is based on three modules: *Agent Manager*, *Environment Manger* and *Event Layer* that we've already detailed in the Simulator chapter.

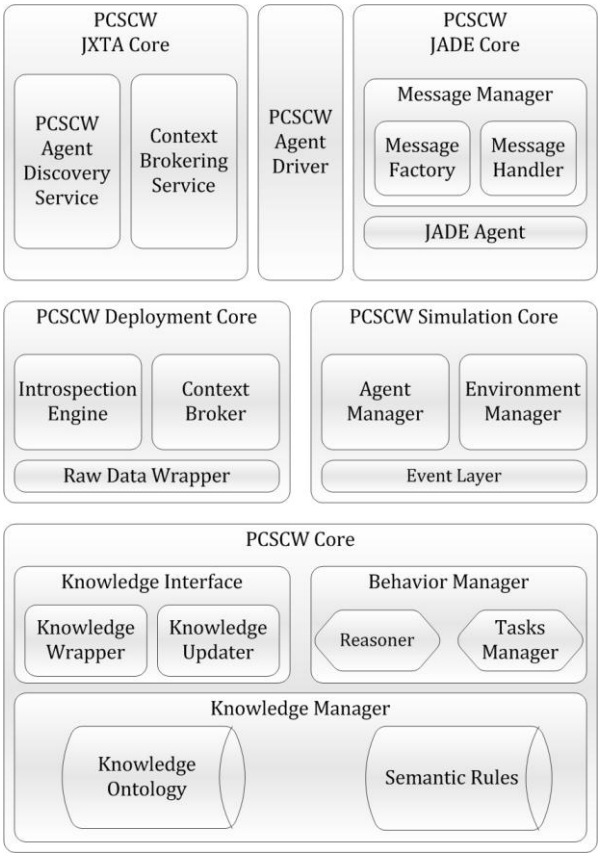

Figure 68 PCSCW Hybrid Agent Architecture

- *PCSCW JADE Core*: laying on top of the *simulation core* this part of the architecture represents how the JADE Framework is integrated to the simulator. It has not been integrated directly inside the simulation core as we want to leave us the possibility to implement a different version of the simulator using another framework or even a homemade multi-agent system.
- *PCSCW JXTA Core*: this "top" level layer contains JXTA services to facilitate the collaboration of devices and provide simple way to exchange context information. We've got two main services at this level:
  - *Context Brokering Service*: a service to simplify access to agent's context information and retrieval, it relies on the *deployment core* and improve it;
  - *PCSCW Agent Discovery Service*: similar to classical agent discovery, this module enables a device to find other PCSCW Agents (devices) in order to be able to

cooperate with them; this specific part of the layer is particularly interesting as it can bring new devices and then new agents in the simulation.

- *PCSCW Agent Driver*: to enable the hybrid architecture we do not simply need to integrate real devices information and knowledge to the simulation, but we also need to allow real and simulated agents to communicate and interact. For us the natural way to do it is to integrate deployment and simulation features. Then, each real device in addition to the *PCSCW deployment core* has the PCSCW *simulation core*. As we described earlier in our implementation of the simulation we have used the JADE Framework; then to be able to communicate with other JADE agents (the simulated ones) we decided that each devices taking part in the simulation would have to use its own part of the JADE framework. In this perspective, a device will need to create its own JADE agent and make it communicate with the rest of the simulation. This can be done thanks to the capacity of JADE to interconnect multiple agents' containers. Still it is not sufficient to ensure a proper functioning and interactions of all agents in a scenario. Indeed, we still require making the real agent and its simulated alter ego interact, that's why we introduce the notion of *Agent Driver*. This part of the architecture provides to a real agent the necessary and sufficient methods and services to manipulate its JADE representation. This module is critical as the JADE agent is the entry point of the device to the simulation and has to offer it the possibility to communicate with the simulated environment through the event layer and other agents through the messaging system.

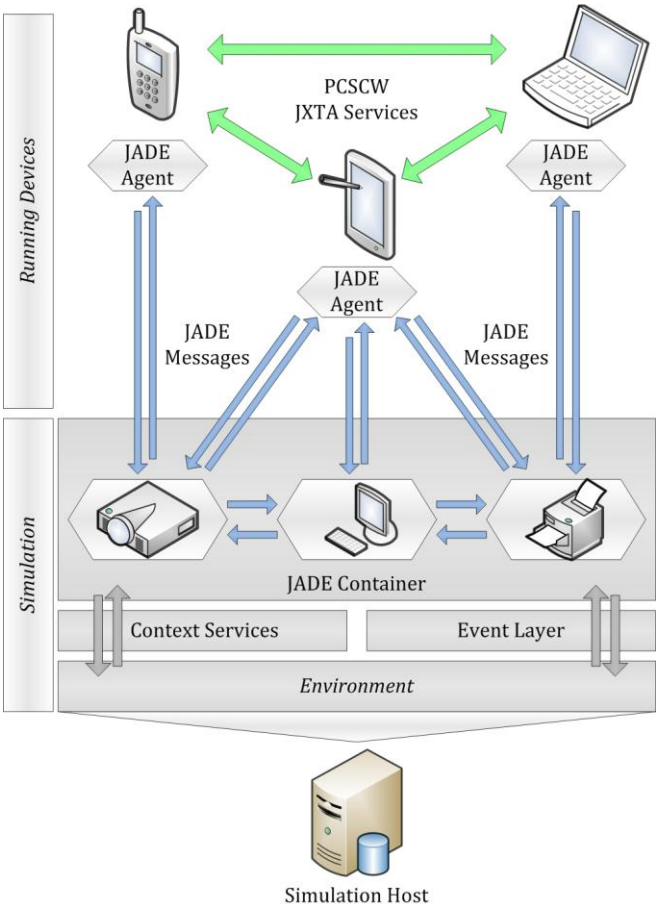

Figure 69 Agent's Interactions

Figure 69 summarizes and depicts how the various agents interact. At the bottom of the figure we've got the machine hosting the simulation, with the previously detailed environment and the JADE agents' container.

Interactions between these two parts of the simulation are ensured by the Event Layer and the Context Services. Interactions between JADE agents are provided by the JADE agent messaging system and the message manager of each agent. Interactions between "real" devices are channel through JXTA services. Finally, interactions between simulated and real devices are performed by making real devices driven agents interact with pure simulated ones.

From the pervasive computing perspective, the hybrid architecture we propose stands on the border of two worlds, simulated environments and real deployed applications. By putting a bridge between them we try to make them benefit from each other. We know that pervasive computing is a recent domain and we can't really imagine what "computing" will mean in a few decades. At present time smart devices are poorly linked and it almost always requires a human intervention. Hence developing new ways to make smart devices interact and cooperate automatically is a more and more interesting and promising domain. Considering this aspect our architecture proposes a real framework to develop such behaviours. Going a little further, an important point to notice is a new perspective brought to the simulator by the introduction of real devices. Indeed in the case of pure simulations, all types of agents are already known and their possible actions and constraints are well defined. By enabling the possibility to interact with any kind of real device we extend the functioning of the simulator to make it accept completely the open world assumption. Hence, it is necessary for the simulation *Environment* and simulated agents to be able to update their knowledge and handle unexpected agents, resources and behaviours. Obviously we can't pretend that simulated agents will fully interact with real ones as there are still some insurmountable issues when you deal with a completely open world. However a really interesting perspective we found during our work on this architecture could give us a way to have a better management of unexpected and "exotic" devices. This perspective is the possible extension of the system to capture and trace behaviour of a real device using the PCSCW model. This acquisition would provide us rich and contextualized data from which we would be able to extract knowledge about devices interactions and behaviours in pervasive computing environment. Thereby we should be able to integrate some of the new behaviours of these devices in our simulations. In addition to this integration of new behaviours and concepts, it would provide us useful information to refine, enhance and enrich the quality of simulation scenarios and simulated agents.

On the long range we can even imagine a more sophisticated system which could do more than just manage and simulate new devices behaviour. Indeed we think that the analysis of new behaviours put in perspective with the PCSCW model could give us key information to allow the dynamic creation of new behaviours for devices. Besides, the hybrid architecture we propose would be a reliable and efficient way for devices to "automatically" evaluate the efficiency of generated behaviours by running their own simulations without the intervention of humans. This would be even more efficient with the perspective of creating device collaboration building blocks, which could then be automatically assembled by the system according its experience and feedbacks, and directly tested with the embedded simulator. In conclusion, if we think a little further ahead, this work could be a small brick toward the real breakthrough in ambient intelligence, where devices would be able to seamlessly communicate, interact, collaborate, learn from their context, learn from each other and optimally adapt their behaviour.

Publications derived from this research

1. Hamadache, K., Lancieri, L.; Role-Based Collaboration Extended to Pervasive Computing. In: Intelligent Networking, Collaborative Systems and Applications. Berlin, Germany: Springer-Verlag. Series "Studies in Computational Intelligence". In process. (2010)

2. Hamadache, K., Lancieri, L.; Towards Ontological Model Accuracy's Scalability: Application to the Pervasive Computer Supported Collaborative Work, International Conference on Machine and Web Intelligence (**ICMWI 2010**), October, Algiers, Algeria, pp. (2010)

3. Hamadache, K., Lancieri, L.; Dealing with Device Collaboration Rules for the PCSCW Model. In: 16th Collaboration Researchers' International Working Groupware Conference on Collaboration and Technology **(CRIWG 2010)**, September, Maastricht, the Netherlands, pp. (2010).

4. Hamadache, K., Lancieri, L.; Hybrid Architecture for Pervasive Computing Supported Collaborative Work Application and Simulation. In: 19th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises **(WETICE 2010**), June, Larissa, Greece, pp. (2010)

5. Hamadache, K., Lancieri, L.; Indicateurs pour l'évaluation du Travail Collaboratif en Environnement Pervasif. In: 2nd French Workshop Atelier Interaction, Contextes, traces **(ICT 2010)**, May, Marseille, France, pp. (2010)

6. Hamadache, K., Lancieri, L.; Role-Based Collaboration Extended to Pervasive Computing. In: International Conference on Intelligent Networking and Collaborative Systems (**INCoS 2009**), November, Barcelona, Spain, pp. 9 - 15 (2009)

7. Hamadache, K.; Lancieri, L.; Strategies and Taxonomy, Tailoring your CSCW Evaluation. In: 15th Collaboration Researchers' International Workshop on Groupware (**CRIWG 2009**), September, Peso da Régua, Portugal, pp. 206 - 221 (2009)

8. Hamadache, K., P. Manson, P., Lancieri, L.: Pervasive services, brainstorming in situation of mobility. In: 3rd International Conference on Pervasive Computing and Applications **(ICPCA 2008)**. October, Alexandria, Egypt, pp. 709 - 714 (2008)

References

[Ahn et al, 2003] Ahn, G.J., Zhang, L., Shin, D., Chu, B. : Authorization Management for Role-Based Collaboration. In: *IEEE International Conference on Systems, Man and Cybernetics*, pp. 4128-4134 (2003).

[Antunes et al, 2003] Antunes, P. and Costa, C. 2003. :Perceived value: A low-cost approach to evaluate meetingware. In: Favela, J., Decouchant, D. (eds.) CRIWG 2003. LNCS, vol. 2806, pp. 109–125. Springer, Heidelberg (2003)

[Antunes et al, 2005] Antunes, P., Borges, M.R.S., Pino, J.A. and Carriço, L. :Analytic Evaluation of Groupware Design. 9th international conference, CSCWD 2005, Coventry, UK, May 24-26, (2005)

[Antunes et al, 2006] Antunes, P., Ferreira, A. and Pino, J.: Analyzing shared workspaces design with human-performance models. In: Dimitriadis, Y.A., Zigurs, I., and Gómez-Sánchez, E. (eds.) CRIWG 2006. LNCS, vol. 4154, pp. 62–77. Springer, Heidelberg (2006).

[Armstrong, 2001] Armstrong, J.S. :Principles of Forecasting: a Handbook for researchers and Practitioners. Kluwer Academic Publishers,(2001).

[Avouris et al, 2003] Avouris, N., Komis, V., Fiotakis, F., Margaritis, M., Tselios, N. :On tools for analysis of collaborative problem solving. Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies. (2003)

[Bachimont, 2000] Bachimont, B.: Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en Ingénierie des connaissances. In J. Charlet, M. Zacklad, G. Kassel & D. Bourigault (Eds.), Ingénierie des connaissances, évolutions récentes et nouveaux défis. Paris: Eyrolles. (French) (2000).

[Baeza-yates and pino, 1997] Baeza-Yates, R. and Pino, J.A. A First Step to Formally Evaluate Collaborative Work. In Proceedings of Group 97, Phoenix, USA (1997).

[Baeza-yates and pino, 2006] Baeza-Yates, R. and Pino, J. Towards formal evaluation of collaborative work and its application to information retrieval. Information Research 11(4), 271 (2006).

[Baker et al, 2001] Baker, K., Greenberg, S. and Gutwin, C. Heuristic Evaluation of Groupware Based on the Mechanics of Collaboration. Proceedings of the 8th IFIP Working Conference on Engineering for Human-Computer Interaction (EHCI'01). (May 11-13, Toronto, Canada) (2001).

[Baker et al, 2002] Baker, K., Greenberg, S. and Gutwin, C. :Empirical development of a heuristic evaluation methodology for shared workspace groupware. In: CSCW ’02, pp. 96–105 (2002)

[Bannon, 1996] Bannon, L. :Use, design and evaluation: steps towards an integration. In: Shapiro, D.,M. Tauber and R. Traunmuller, Eds., The design of computer supported cooperative work and groupware systems. Elsevier, Amsterdam, 1996. 423-443, (1996)

[Barbuceanu et al, 1998] Barbuceanu, M., Gray, M., Mankovski, S. : Coordinating with Obligations. In: *Agents ’98*, Minneapolis, May, 1998, pp. 62 – 69, (1998)

[Bardram and Hansen, 2004] Bardram, J. E., Hansen T. R. :The AWARE Arhcitecture: Supporting Context-Mediated Social Awanreness in Mobile Cooperation”. Proceedings of the 2004 ACM conference on Computer supported cooperative work, Chicago, Illinois, USA, pages: 192 – 201, (2004)

[Benford and Fahlen, 1993] Benford, S. D., Fahlen, L. E.: "A spatial model of interaction in large virtual environments". In: Proceedings of the 3rd European Conference on CSCW (ECSCW’93). Milan, Italy: Kluwer Academic Publishers, 1993. 109-124, (1993)

[Buszko et al, 2001] D. Buszko, W.-H. Lee, and A. Helal, "Decentralized ad hoc groupware API and framework for mobile collaboration". In Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work, Boulder, USA, (2001).

[Caporuscio and Inveraldi, 2003] Caporuscio., M and .Inverardi L. :Yet Another Framework for supporting Mobile and Collaborative Work”, Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Entreprise (WETICE’O3), Linz, Austria, June (2003).

[Caporuscio et al, 2003] Caporuscio. M, Carzaniga. A, and Wolf. A.L. :Design and evaluation of a support service for mobile, wireless publish/subscribe applications. Technical Report CU-CS-944-03, Department of Computer Science, University of Colorado, Jan. 2003. Available at http://www.di.univaq.it/~caporusc/mobikit. (2003).

[Card et al, 1980] Card, S.K, Moran, T.P, Newell and A. 1980 .The Keystroke-level model for user performance time with interactive systems. Communications of the ACM 23 (7) (1980) 396-410 (1980).

[Card et al, 1983] Card, S., Moran, T., Newell, A.: The Psychology of Human-Computer Interaction. Lawrance Elrbaum, Hillsdale, NJ (1983)

[Carrol, 1995] Carroll, J.M. :Scenario-based design: envisoning work and technology in system development, New York, (1995).

[Carroll et al, 2006] Carroll, J.M., Rosson, M.B., Convertino, G. and Ganoe, C.H. Awareness and teamwork in Computer-Supported Collaborations. Interacting with Computers, 18(1), 21-46, (2006).

[Carzaniga et al, 2000] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 219–227, Portland, OR, July (2000).

[Cerratto, 1999] T. Cerratto "Activité collaborative sur réseau, une approche instrumentale de l'écriture en collaboration". Thèse de doctorat, université Paris VIII, France (1999).

[Chang et al, 2006] Chang, C.K, Zhang, J. and Chang, K.H. :Survey of computer-supported collaboration in support of business processes", Int. J. Business Process Integration and Management, Vol. 1, No. 2, pp.76-100, (2006).

[Checkland, 1989] Checkland, P. :Soft Systems Methodology. Human Systems Management. 8(4):273-289, (1989).

[Cheverst et al, 1999] K. Cheverst, G. Blair, N. Davies, and A. Friday. :The report of mobile-awareness in collaborative groupware". Personal Technologies, 3(1-2):33 42, (1999).

[Clement and Van Besselaar, 1993] Clement, A and Van Besselaar, P. :Participatory design projects: A retrospective look. Communications of the ACM, 36(6), 19-27 (1993).

[Cohen, 1996] Cohen, J. :Computer mediated communication and publication productivity among faculty", Internet Research Electronic Networking applications and policy, Vol. 6, No.23, pp.41-63 (1996).

[Damianos et al, 1999] Damianos, L., Hirschman, L., Kozierok, R., Kurtz, J., Greenberg, A., Holgado, R., Walls, K., Laskowski, S. and Scholtz, K. :Evaluation for collaborative systems. ACM Computing Surveys, 31(2),(1999).

[Daneshgar and Ray, 2000] Daneshgar F, Ray P. :Awareness modeling and its application in cooperative network management In: Proceedings of the 7th IEEE International Conference on Parallel and Distributed Systems. Iwate, Japan: IEEE Press, 2000. 357-363 (2000).

[David et al, 2007] David, B., Chalon, R., Delotte, O., Masserey, G. :ORCHESTRA: formalism to express static and dynamic model of mobile collaborative activities and associated patterns, HCI International 2007 Beijing 22-27 July 2007 In J. Jacko (Ed.): Human- Computer Interaction, Part I, HCII 2007, LNCS 4550, pp. 1082–1091, 2007. © Springer Verlag Berlin Heidelberg (2007)

[Dillenbourg et al, 1994] Dillenbourg, P.; Mendelsohn, P.; and Schneider, D. :The distribution of pedagogical roles in a multi-agent learning environment. In R.Lewis, and P.Mendelsohn., eds., *Lessons from Learning*. North-Holland. 199—216 (1994)

[Dillenbourg, 1999] Dillenbourg P. :What do you mean by collaborative learning?" In P. Dillenbourg (Ed) Collaborative Learning: Cognitive and Computational Approaches, Oxford: Elsevier, 1-19, (1999)

[Dourish and Belloti, 1992] P. Dourish and V. Bellotti, "Awareness and Coordination in Shared Workspaces," Proc. ACM Conf. Computer-Supported Cooperative Work, pp. 107-114, Nov. (1992).

[Dourish and Bellotti, 1992] P. Dourish and V. Bellotti. :Awareness and coordination in shared workspaces", Proceedings of ACM conference on Computer Supported Cooperative Work, Toronto, Canada, Nov. 1992, pp.107-114 (1992).

[Drury et al, 1999] Drury, J., Damianos, L., Fanderclai, T., Kurtz, J., Hirschman, L. and Linton F. 1999. Methodology for Evaluation of Collaborative Systems. (1999).

[Dugan et al, 2002] Dugan, R.F.Jr., Glinert, E.P. and Rogers, E.H. : CAMELOT: Technology Focused Testing of CSCW Applications, System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference. 6-9 Jan 2003 Page(s):10 pp, (2002)

[Dugan, 2000] Dugan, R.F. A testing methodology and architecture for Computer Supported Cooperative Work software. Doctoral thesis, Rensselaer Polytechnic Institute, Department of Computer Science, (2000).

[Dumas and Redish, 1993] Dumas, J.S. and Redish, J.C. :A practical guide to usability testing. Ablex, (1993).

[Duque et al, 2008] Duque, R., Gallardo, J., Bravo, C. and Mendes A. J. : Defining Tasks, Domains and Converstaional Acts in CSCW Systems : the SPACE-DESIGN Case Study, Journal of Universal Computer Science, vol. 14, no. 9, 1463-1479, (2008).

[Dustdar and Gall, 2003] Dustdar, S and Gall, H, "Architectural concerns in distributed and mobile collaborative systems", 11th Euromicro conference on Parallel, Distributed and Network-Based Processing, Feb, 05-07, genoa, Italy, (2003).

[Edwards, 1996] Edwards, W.K. : Policies and Roles in Collaborative Applications'. In: *Proceedings ACM Conference Computer Supported Cooperative Work*, Cambridge, MA, pp. 11—20, (1996).

[Ellis et al, 1991] Ellis, C.A, Gibbs, S.J. and Rein, G.L. :Groupware: some issues and experiences", Communications of ACM, Vol. 34, No. 1, pp. 38-58, (1991).

[Elmasri et al, 2007] Elmasri, R, Fu, J and Li, F. :Multi-level Conceptual Modeling for Biomedical Data and Ontologies Integration" CBMS '07. Twentieth IEEE International Symposium on Computer-Based Medical Systems. Volume , Issue , 20-22 June Pages:589 – 594, (2007).

[Engelbart and English, 1968] Engelbart, D and English, W. :A Research center for augmenting human intellect". In Proc. The Fall Joint Computing Conference, vol.33, pp. 395-410, (1968).

[Ferreira and Antunes, 2005] Ferreira, A. and Antunes, P. 2005. Analytical Evaluation of groupware usability in Concerted Scenarios. http://docs.di.fc.ul.pt/jspui/bitstream/10455/3026/1/05-15.pdf, (2005)

[Fetter and Gross, 2007] fetter.M and Gross. T. :time, space, connection: scaling ambient intelligence", UPGRADE- the European Journal for the Informatics Professional VIII, vol. 4, pp.44-49, (2007).

[Flower and Hayes, 1978] Flower, L and Hayes, J.R.. :A cognitive process theory of writing", *College Composition and Communication*, 1978, Vol. 32, pp. 365-387, (1978).

[Gallardo et al. 2007] Gallardo, J. Bravo, C.: Coordination and Communication Protocols for Synchronous Groupware: A Formal Approach. In: D. Cunliffe (ed.): Proceedings of the Second IASTED Conference on Human-Computer Interaction. ACTA Press, 2007, 55-61, (2007).

[Gao-feng et al, 2006] Gao-feng, J, Yong, T, Ling-kun, W, Miao-miao, C. :a Service-Oriented Group Awareness Model". In 1st International Symposium on Pervasive Computing, pp. 635-640 (2006)

[Garcia et al, 2000] Garcia, O, Favela, J and Machorro, R,. :Amotional Awareness in Collaborative Systems", (2000)

[Greenbaum and Kyng, 1991] Greenbaum, J and Kyng, M. :Design at work: Cooperative design of computer systems" Hillsdale, NJ: Lawrence erlbaum Associates, (1991).

[Greif et al. 1988] Greif, I. : Computer-Supported Cooperative Work: A Book of Readings. Morgan Kaufmann, San Mateo CA (1988).

[Griswold et al, 2003] Griswold.W.G, Boyer.R, Brown.S.W, and Tan Minh Truong,. :A component architecture for an extensible, highly integrated context-aware computing infrastructure". In Proc. of the 25th International Conference on Software Engineering (ICSE 2003), Portland, Oregon, (2003).

[Gross, 2008] Gross, T. :Cooperative ambient intelligence: towards autonomous and adaptive cooperative ubiquitous environments" Int. J. Autonomous and Adaptive Communications Systems, Vol. 1, No. 2, (2008).

[Gruber, 1993] Gruber, T.R.. : A translation approach to portable ontology specifications, Knowledge Acquisition 5, 199-220, (1993).

[Grudin, 1994] Grudin, J. :Groupware and Social Dynamics: Eight Challenges for Developers. In Communications of the ACM, volume 37, number 1. ACM press. pp. 92-105, (1994).

[Grudin, 1994] Grudin,J. : Computer-Supported Cooperative Work: History and Focus. Computer 27(5), 19-26, (1994).

[Grudin, 1994] J. Grudin. :Computer-Supported Cooperative Work: its history and participation". Computer 27(5): 19-26, (1994).

[Grudin, 2001] Grudin, J. :Partitioning Digital Worlds-Focal and Peripheral Awareness in Multiple Monitor Use," Proc. ACM Conf. Human Factors in Computing Systems, pp. 458-465, (2001).

[Grudin, 2001] Izadi, S, Coutinho, P, Rodden, T, and Smith, G. :The FUSE Platform: Supporting Ubiquitous Collaboration within Diverse Mobile Environments," Automated Software Eng., vol. 9, no. 2, pp. 167-186, (2002).

[Gutwin and Greenberg, 2000] Gutwin, C. and Greenberg, S. :The mechanics of collaboration: Developing low cost usability evaluation methods for shared workspaces. In: WETICE '00, pp. 98–103. IEEE Comp. Soc, Los Alamitos (2000).

[Gutwin and Greenberg, 2002] Gutwin, C and Greenberg, S. :A Descriptive Framework of Workspace Awareness for Real-Time Groupware," Computer Supported Cooperative Work, vol. 11, nos. 3/4, pp. 411-446, (2002).

[Hamadache et al, 2007] Hamadache, K.; Bertin, M.; Bouchacourt, A.; Benyahia, I; :Context-aware communication services: an ontology based approach", 2nd International Conference on Digital Information Management (ICDIM'07). October, Lyon, France, (2007).

[Hartwood and Procter, 2000] Hartswood, M. and Procter, R. 2000. :Design guidelines for dealing with breakdowns and repairs in collaborative work settings. International Journal of Human- Computer Studies, 53, 91-120, (2000).

[Haynes et al, 2004] Haynes, S., Purao, S. and Skattebo, A. :Situating evaluation in scenarios of use. Proceedings of CSCW '04, November 6-10, Chicago, Illinois, USA. ACM. 92-101, (2004).

[Helms et al, 2000] Helms, J., Neale, D.C. and Caroll, J.M. :Data logging: Higher-level capturing and multi-level abstracting of user activities. In Proceedings of the 40th annual Meeting of the Human Factors and Ergonomics Society (pp.303-306). Santa Monica, CA: human Factors and Ergonomics Society, (2000).

[Herskovic et al, 2007] Herskovic, V., Pino, J.A., Ochoa, S.F. and Antunes, P. : Evaluation methods for groupware systems. J.M. Haake, S.F. Ochoa, and A. Cechich (Eds.): CRIWG 2007, LNCS 4715, pp. 328–336, 2007. © Springer-Verlag Berlin Heidelberg, (2007).

[Herskovic et al, 2007] Herskovic, V., Pino, J.A., Ochoa, S.F., Antunes, P.: Evaluation Methods for Groupware Systems. In: J.M. Haake, S.F. Ochoa, and A. Cechich (Eds.): CRIWG 2007, LNCS 4715, pp. 328–336. © Springer-Verlag Berlin Heidelberg. (2007).

[Howard, 1988] Howard, R. :Panel remarks: CSCW: What does it mean?. Proceedings of CSCW'88. Portland, OR: Association for Computing Machinery, (1988).

[Hu et al, 2007] Hu, C, Liu, Q, Liu, X, Liao, C, and McEvoy, P. :POEMS: a paper based meeting service management tool, (2007).

[Huang, 2003] Huang, J.P.H. :An evaluation framework to support development of virtual enterprises. Electronic Journal of Information systems Evaluation Volume 6 Issue 2, (2003).

[Hughes et al, 1991] Hughes, Randall, J.D and Shapiro, D. :CSCW: Discipline or paradigm?, in L. Bannon, M. Robinson, and K. Schmidt, (Eds). Proceedings of the Second European Conference on CSCW- ECSCW'91, (pp. 309-323). Dordrecht: Kluwer Academic Publishers, (1991).

[Hughes et al, 1994] Hughes, J., King, V., Rodden, T. and Andersen, H. :Moving out from the control room: Ethnography in system design. In: CSCW '94, pp. 429–439. ACM Press, New York, (1994).

[Huis in't veld et al, 2003] Huis in't Veld, M., Andriessen, J. and Verburg, R. :E-magine: The development of an evaluation method to assess groupware applications. In: WETICE '03, vol. 153, IEEE, Los Alamitos (2003).

[Iqbal et al, 2005] Iqbal, Sturm, R..J, Kulyk, O, Wang, J and Terken, J. :User-centred design and evaluation of ubiquitous services. Proceedings of the 23rd annual International conference in Design of Communication: documenting & designing for pervasive information, (2005).

[Jianshe and Liyi, 2003] Jianshe, Z and Liyi, W. :Computer supported collaborative editing system with speech-recognition functionality, 8th European conference of Computer supported collaborative Work, Helsinki, (2003).

[Johansen 1988] Jonhansen,R. .: Groupware: Computer Support for business teams. The Free Press, New York, (1988).

[Kieras, 1996] Kieras, D.: A Guide to Goms Model Usability Evaluation Using Ngomsl. University of

Michigan, (1996).

[Kirda et al, 2002] Kirda, E, Fenkam, P, Reif,, G and Gall, H. :A service architecture for mobile teamwork". In Proceedings of the 14th international conference on Software engineering and knowledge engineering, Ischia, Italy, (2002).

[Kirda et al, 2003] Engin Kirda, *et* al. :A Service Architecture for Mobile Teamwork", Intl. Journal of Software Engineering and Knowledge Engineering, Vol. 13, No. 4 (2003) 447467, (2003).

[Kling, 1991] R. Kling. :Cooperation, coordination, and control in computer-supported work". Communications of the ACM, 34 (12), 83-88, (1991).

[Kyung-Kim and Kim, 2006] Kim, M.K and Kim, H.C. :Awareness and Privacy in Groupware systems. Computer Supported Cooperative Work in Design, 1-5, (2006).

[Lancieri et al, 2005] Lancieri, L, Lavallard, A and Manson, P. :E-Brainstorming: Optimization of Collaborative Learning Thanks to Online Questionnaires, In IADIS International Conference Cognition and Exploratory Learning in Digital Age (CELDA 2005), Porto, Portugal,(2005).

[Lancieri, 2004] Lancieri, L. :Reusing Implicit Cooperation, a Novel approach to knowledge management"; TripleC. International Journal, 2004 pp 28-46; ISSN 1726-670X, (2004).

[Lau et al, 2003] Lau, L.M.S., Adams, C.A., Dew, P.M, Leigh and C. :Use of scenario evaluation in preparation for deployment of a collaborative system for knowledge transfer. Proceedings of the 12th IEEE International Workshops on Enabling Technologies (WETICE 2003): Infrastructure for Collaborative Enterprises. IEEE Computer Society Press. 148-152. (2003).

[Lay and Karis, 1991] Lay, M.M and Karis, W.M. :Collaborative Writing in Industry: Investigations in theory and practice", Baywood publishing Company, Amityville, (1991).

[Lee et al, 2004] Lee, H, Mihailescu, P, and Shepherdson, J. :Multi-agent Technology as an Enabler of Computer Supported Cooperative Work for the Mobile Workforce. M. Klusch et al. (Eds.): CIA 2004, LNAI 3191, pp. 183–198, 2004. © Springer-Verlag Berlin Heidelberg , (2004).

[Liccardi et al, 2007] Liccardi, I, Davis, H.C and White,S. :CAWS: a wiki system to improve workspace awareness to advance effectiveness of co-authoring activities". CHI 2007, San Jose, California, USA, (2007).

[Lipponen, 2002] Lipponen, L. :Exploring Foundations for Compute Supported collaborative Learning" Proceedings of Computer Supported Collaborative Learning 2002, pp. 72-81, (2002).

[Litiu and Prakash, 2000] Litiu, R and Prakash, A. :Developing Adaptive Groupware Applications Using a Mobile Component Framework," Proc. ACM Conf. Computer Supported Cooperative Work, pp. 107-116, (2000).

[Malone and Crowston, 1990] Malone, T.W. and Crowston, K. :What is coordination theory and how can it help design cooperative work systems", Proceedings of the ACM Conference on Computer Supported Cooperative work (CSCw), pp. 357-370, (1990).

[Marques et al, 2006] Marques, J. M., Navarro, L. LaCOLLA: A Middleware to Support Self-sufficient Collaborative Groups. Computers and Artificial Intelligence 25(6).(2006).

[McDonald et al, 2004] McDonald, D.W., Weng, C., Gennari, J.H. :the multiple views of inter-organizational Authoring, CSCW, ACM, Chicago, IL, pp. 594-573, (2004).

[McKinley and Li, 1999] McKinley, P.K and Li, J. :Pocket Pavilion : A Synchronous Collaborative Browsing Application for Wireless Handheld Computers", Tech. Report - MSU-CPS-99-38, Dept. of Computer Science and En& Michigan State University, Michigan, (1999).

[McManus, 1997] McManus, M. M. :Computer Supported Collaborative Learning" Special issue: enterprise modelling: case studies and business process re-engineering Pages: 7 – 9, (1997).

[Meier and Cahil, 2003] Meier, R and Cahil, V. :Exploiting proximity in event-based middleware for collaborative mobile applications". In 4th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'03), Paris, France, (2003).

[Mendes de Arauja et al, 2002] Mendes de Arauja, R., Santoro and F.M., Borges, M.RS. :The CSCW Lab for Groupware Evaluation. J.M. Haake and J.A. Pino (Eds.): CRIWG 2002, LNCS 2440, pp. 222–231, 2002. © Springer-Verlag Berlin Heidelberg , (2002).

[Miles and Snow, 1986] Miles, R.E and Snow, C.C. :Network organizations: New Concepts for New Forms, California Management Review No. 28, Haas School of Business at the University of California, Berkeley, 1986, pp. 62-78, (1986).

[Milewski and Smith] Milewski, A and Smith, T. :Providing Presence Cues to Telephone Users, Proc. ACM Conf. Computer Supported Cooperative Work, pp. 89-96, (2000).

[Munson and Dewan, 1997] Munson, J and Dewan, P. :Sync: a Java framework for mobile collaborative applications. IEEE Computer, 30(6):59 66, (1997).

[Neale et al, 2004] Neale, D. C., Carroll, J.M., Rosson, M. B.: Evaluating Computer-Supported Cooperative Work: Models and Frameworks. In: CSCW'04, Chicago, Illinois, USA. (2004)

[Nicollin and Sifakis, 1994] Nicollin, X, and Sifakis, J. :the algebra of timed processes, ATP: Theory and Application Information and Computation, Vol. 114, pp. 131-178, (1994).

[Nielsen and Molich, 1990] Nielsen, J. and Molich, R. :Heuristic evaluation of user interfaces. Proc ACM CHI'90, 249-256, (1990).

[Nohria and Berkley, 1994] Nohria, N. and Berkley, J.D. :The Virtual Organization: Bureaucracy, Technology, and the Implosion of Control, The post-Bureaucratic Oraganization: New Perspectives on Organizational Change, in C. Heckscher and A. Donnelson (Eds), Sage , Thousand Oaks, CA, PP.108-128, (1994).

[Olson and Olson, 2001] Olson, G.M. and Olson, J.S. :Technology support for collaborative workgroups. In G.M. Olson, T.W. Malone, and J.B. Smith (Eds), Coordination theory and collaboration technology (pp. 559-584). N.J. Mahwah: Lawrence Erlbaum, (2001).

[Paternò, 1999] Paternò,F. : Model-based Design and Evaluation of Interactive Applications. In: Paternò, F.(ed.) Model-based Design and Evaluations of Interactive Applications, Springer, Heidelberg (1999).

[Penichet et al, 2007] Penichet, V.M.R, Lozano, M, Gallud, J.A and Tesoriero, R. : Task Modelling for Collaborative Systems, in Task Models and Diagrams for User Interface Design", pages 287-292, Springer Berlin / Heidelberg, (2007).

[Pinelle and Gutwin, 2000] Pinelle, D. and Gutwin, C. :A Review of Groupware Evaluations, Proceedings of Ninth IEEE WETICE 2000 Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Gaithersburg, Maryland, June, (2000).

[Pinelle and Gutwin, 2002] Pinelle, D. and Gutwin, C. :Groupware walkthrough: adding context to groupware usability evaluation. In: CHI '02, pp. 455–462. ACM Press, New York, (2002).

[Pinelle et al, 2003] Pinelle, D., Gutwin, C. and Greenberg, A. :Task analysis for groupware usibility evaluation: modelling shared-workspace tasks with the mechanics of collaboration. Transactions on Computer human Interaction, 10(4), 281-311, (2003).

[Pinsonneault and Kraemer, 1989] Pinsonneault, A. and Kraemer, K.L. :The impact of technological support on groups: an assessment of the empirical research. Decision Support Systems, 5(2), 197-211, (1989).

[Polson et al, 1992] Polson, P., Lewis, C., Rieman, J. and Wharton, C. :Cognitive Walkthroughs: a method for theory-based evaluation of user interfaces. International Journal of Man-Machine Studies, 36, 741-773, (1992).

[Poltrock et al. 1999] Poltrock, S., Grudin, J. : CSCW, groupware and workflow: experiences, state of the art, and futures trends. In: CHI 1999 Extended Abstracts on Human Factors in Computing Systems, pp. 120-121. ACM Press, New York, (1999).

[Poltrock et al. 2005] Poltrock, S., Grudin, J. : Computer Supported Cooperative Work and Groupware (CSCW). In: Costabile, M.F., Paternò, F.(eds.) INTERACT 2005. LNCS, vol. 3585, Springer, Heidelberg (2005).

[Ramage, 1995] Ramage, M. :Evaluation of Cooperative Systems: First-year Report of PhD. http://www.comp.lancs.ac.uk/computing/research/cseg/projects/evaluation/1YR_contents.html, (1995).

[Ramage, 1996] Ramage, M. :CSCW Evaluation in Five Types. Report CSEG/17/96, Computing Department, Lancaster University, UK, (1996).

[Ramage, 1999] Ramage, M. :How to Evaluate Cooperative Systems, Computing Department, University of Lancaster, Lancaster, United Kingdom, Ph.D Thesis, (1999).

[Randall et al, 1996] Randall, D., Twidale, M. and Bentley, R. 1996. Dealing with Uncertainty-Perspectives on the Evaluation Process. In: Thomas, P.J. (ed.), CSCW Requirements and Evaluation, Springer, (1996).

[Reiff-Marganiec et al, 2008] Reiff-Marganiec, S.T, Truong, H, Casella, G, Dorn, C, Dustdar, S, Moretzky, S. :The inContext Pervasive Collaboration Services Architecture. 1st. European Conference, Service Wave 2008, Madrid, Spain; 10.12.2008 - 13.12.2008; Springer, LNCS 5377 (2008), ISBN: 978-3-540-898962; S. 134 – 146, (2008).

[Roberts, 2004] Roberts, T.S. :Online collaborative learning: theory and practice". Idea group publishing, (2004).

[Rodden, 1996] Rodden, T.: Populating the application: A model of awareness for cooperative applications. In: Proceedings of the ACM CSCW'96 Conference on Computer Supported Cooperative Work. Boston, MA: ACM Press, 87-96, (1996).

[Ross et al, 1995] Ross, S., Ramage, M. and Rogers, Y. :PETRA: Participatory evaluation through redesign and analysis. Interacting with Computers, 7(4), 335-360, (1995).

[Rosson and Carroll, 2003] Rosson, M.B. and Carroll, J.M. :Scenario-based design. J.A. Jacko & A. Sears, eds. The Human-Computer Interaction Handbook. Lawrence Erlbaum Associations, 1032-1050, (2003).

[Roth and Unger, 2001] Roth, J and Unger, C, :Using Handheld Devices in Synchronous Collaborative Scenarios, Personal and Ubiquitous Computing, Vol. 5, No, 4, London,243-252, (2001).

[Sacramento et al, 2004] Sacramento, V, Endler, M, Hana K. Rubinsztejn, Luciana S. Lima, Kleder. Gonçalves, Nascimento, F.N and Bueno, G.A. :MoCa: a middleware for developing collaborative applications for mobile users", in IEEE Distributed systems online, (2004).

[Schmidt, 2002] Schmidt, K. :the problem with awareness: introductory remarks on awareness in CSCW. Computer Supported Cooperative work, vol.11, n°3-4, (2002).

[Schnase et al, 1995] Schnase, J.L, Cunnius, E.L, and Dowton, S.B. :the studySpace Project: Collaborative Hypermedia in Nomadic Computing Environments. Communications of the ACM, 38(8): 72-3, (1995).

[Sheng and al. 2001] Sheng, G, Jin-Peng, Zongxia, H.D. :A role-based collaboration modeling method. Computer Engineering and Applications, 14-18, (2001).

[Sommerville, 2004] Sommerville, I. :Software Engineering. 7 ed: Pearson Education Limited, pp. 153-156, (2004).

[Stiemerling and Cremers, 1998] Stiemerling, O. and Cremers, A. :The use of cooperation scenarios in the design and evaluation of a CSCW system. IEEE Transact. on Software Engineering 25, 140, (1999).

[Streitz, 2001] Streitz, N. :the role of ubiquitous computing and the disappearing computer for CSCW, 2001. Groupware Proceedings. Seventh International Workshop, (2001).

[Strohmaier and Lindstaedt, 2007] Strohmaier, M and Lindstaedt, S.N. :KnowFlow- A Hybrid Approach to Identifying and Visualizing Distributed Knowledge Work Practices". Proceedings of the 40th Hawaii International Conference on System Sciences, (2007).

[Su et al, 2004] Su, X, Prabhu, B.S, Cheng, C.C and Gradh, R. :Middleware for multimedia mobile collaborative system"**,** Wireless Internet for Mobile Enterprise Consortium, California Univ., Los Angeles, CA, USA; 14-15 May 2004, page(s): 112- 119, (2004).

[Suh et al, 2007] Suh, Y., Lee, C., Song, J., Jung, M. and Hwan Yun, M. :An evaluation framework for the design concepts of tangible interface on new collaborative work support system. J.Jacko (Ed.): Human-Computer Interaction, part II, HCII 2007, LNCS 4551, pp. 1210-1219, (2007).

[Sun et al, 2004] Sun, D., Xia, S., Sun, C., Chen, D. : operational transformation for collaborative word processing, CSCW, ACM, chicago, Il, pp. 437-446, (2004).

[Sushman, 1989] Sushman. :Notes on computer support for cooperative work, ACM Transactions on Office Information Systems, 1(4), 320-328, (1989).

[Terry et al, 1995] Terry, D.B, Theimer, M.M, Petersen, K, and Demer, A.J. :Managing update conflict in Bayou, a Weakly Connected Replicated Storage System". Proceedings of the fifteenth ACM symposium on Operating systems principles, Copper Mountain, CO USA, 172-182, (1995)..

[Thomas, 1996] Thomas, P.J. :CSCW requirements and evaluation. London: Springer, (1996).

[Twidale et al, 1994] Twidale, M., Randall, D. and Bentley, R. :Situated evaluation for cooperative systems. Proceedings of the Conference on Computer Supported Cooperative Work (CSCW'94), Chapel Hill, NC, ACM, (1994).

[Van der Veer and Van Wellie, 2000] Van der Veer, G. and Van Wellie, M. :Task based groupware design: putting theory into practice. In proceedings of the Conference on Designing Interactive Systems, pp.326-337. ACM Press, (2000).

[Van Welie et al, 1999] Van Welie, M., Van der Veer, G.C. and Eliens, A. :Breaking down usability. Proceedings of Interact'99, Edinburgh, Scotland, (1999).

[Vizicaino et al, 2005] Vizicaino, A., Piattini, M., Martinez, M. and Aranda, G. :Evaluating collaborative applications from a knowledge management approach. Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Entreprise (WETICE'05), (2005).

[Wang and Johnson, 1994] Wang, D. and Johnson, P. M. :Experiences with CLARE: A Computer-supported Collaborative Learning Environment. International Journal of Human Computer Studies 41(6), 851-879, (1994)

[Weber et al, 2005] Weber, W., Rabaey, J.M. and Aarts, E. :Ambient Intelligence. New York, NY: Springer-Verlag, (2005).

[Weiser, 1991] Weiser, M. :The Computer for the 21st Century. *Scientific American*, 265(3):94–100, September, (1991).

[Weng and Gennari, 2004] Weng, C., Gennari, J.H. :asynchronous collaborative writing through annotations, CSCW, ACM, Chicago, IL, 2004, pp. 578-581, (2004).

[Whittaker and Schwarz, 1999] Whittaker, S. and Schwarz, H. :Meetings of the board: the impact of scheduling medium on long term group coordination in software development. Computer Supported Cooperative Work, 8, 175-205,(1999).

[Wiberg and Grönlund, 2000] Wiberg, M and Ake Grönlund , A, :Exploring mobile CSCW Five areas of questions for further research" Proceedings of IRIS 23. Laboratorium for Interaction Technology, University of Trollhättan Uddevalla. L. Svensson, U. Snis, C. Sørensen, H. Fägerlind, T. Lindroth, M. Magnusson, C. Östlund (eds.), (2000).

[Wiberg, 2001] Wiberg, M. :Knowledge management in mobile CSCW: Evaluation results of a mobile physical/virtual meeting support system" . In Proceedings of the 34th Hawaii International Conference on System Sciences, (2001).

[Yan and Zeng, 2004] Yan, L and Zeng, J. :A Task-Based Group Awareness Model. The 8th International Conference on Computer Supported Cooperative Work in Design Proceedings. Volume 2, Page(s):90 - 94 Vol.2, (2004).

[Yan, 2009] Yan, L. :Realization of a task-Based Group Awareness Model. In International conference on Computational Intelligence and software Engineering, pp. 1-3, (2009).

[Yang et al, 2000] Yang, Y, Sun, C, Zhang, Y, Jia, X. :Real-Time cooperative editing on the internet, page(s): 18-25, (2000).

[Yenumula Venkataramana Reddy, 2006] Yenumula Venkataramana Reddy. :Pervasive Computing: Implications, Opportunities and Challenges for the Society. 1st International Symposium on Pervasive Computing and Applications, (2006)**.**

[Yokota, 2005] Yokota, Y. :Supporting long-term group meetings with a video database, Proceedings of the International Conference Active Media Technology, (AMT 2005), (2005).

[Zhu and alkins, 2007] Zhu, H and Alkins, R. :a tool for role-based chatting, IEEE International Conference on Systems, Man and Cybernetics. ISIC, (2007).

[Zhu and Tang, 2007] Zhu, J and Tang, Y. :a role-based hierarchical group collaborative awareness model and its implementation. Information Technologies and Applications in Education, 2007. ISITAE '07, (2007).

[Zhu, 2004] Zhu, H. :From WYSIWIS to WISINWIS: role-based collaboration. 2004 IEEE International Conference on Systems, Man and Cybernetics, (2004).

[Zhu, 2006] Zhu.H, and Zhou.M.C. :role-based collaboration and its Kernel mechanisms", IEEE Trans on System, Man and Cybernetics, Part C,(2006).