EVALUATION OF MOBILE SHARED WORKSPACES TO IMPROVE THEIR SUPPORT FOR COLLABORATION

TESIS PARA OPTAR AL GRADO DE DOCTOR EN CIENCIAS
MENCIÓN COMPUTACIÓN

VALERIA PAZ HERSKOVIC MAIDA

PROFESORES GUÍA:
JOSÉ A. PINO URTUBIA
SERGIO F. OCHOA DELORENZI

MIEMBROS DE LA COMISIÓN:
JACQUES WAINER
LUIS GUERRERO
ALEXANDRE BERGEL
YADRAN ETEROVIC SOLANO

SANTIAGO DE CHILE
JUNIO 2010
Esta tesis está dedicada a mi familia, y a su pasado, presente y futuro.
Acknowledgements

This work was partially supported by Fondecyt (Chile) grants 11060467 and 1080352, and the Conicyt scholarships for doctoral studies and thesis completion.
I would also like to acknowledge the help of several people whose work is related to this thesis:

- The fire companies of Ñuñoa and Peñalolén (Santiago, Chile) who very kindly allowed me to observe and record their work, and participated enthusiastically in the evaluation processes.
- Emilio Ormeño, who coded the graph visualizer interface.
- Alvaro Monares, Juan Francisco Rodríguez-Covili, Manuel Villarroel, and software engineering students who implemented the MobileMap and ITO applications.
- Antonio Ferreira, David Mejía, and Javier Bustos, for allowing me to participate in and learn from their research.
- Jesús Favela and Pedro Antunes, for generously working with me during my research stays.

I would also like to thank my advisors José A. Pino and Sergio F. Ochoa for their guidance, optimism and support.
Evaluation of Mobile Shared Workspaces to Improve their Support for Collaboration

La evaluación de sistemas colaborativos es una tarea compleja, que no sólo debe examinarse la funcionalidad y usabilidad del sistema, sino también cómo éste apoya el proceso colaborativo. Existen varias dificultades relacionadas con la evaluación de este tipo de sistemas, por ejemplo: la complejidad de los procesos de trabajo que involucran a varios actores y la influencia de factores externos en la colaboración. La reciente aparición de nuevos dispositivos móviles y disponibilidad de redes inalámbricas han aumentado este problema. Pese a que existe la oportunidad de desarrollar sistemas que apoyen la colaboración móvil, pocos métodos de evaluación para esta situación existen, disminuyendo las probabilidades de éxito de los proyectos. Además, se deben considerar aspectos propios de la colaboración móvil como la variabilidad en la composición de los grupos y la flexibilidad de las tareas asignadas.

Esta tesis explora la hipótesis de que un método de evaluación formativo, basado en requisitos funcionales para el apoyo a la colaboración en un contexto específico de trabajo, puede ser usado para medir el apoyo al proceso de colaboración de una aplicación móvil. Este método no considera otros temas importantes, tales como requisitos no funcionales y requisitos que no estén relacionados con la colaboración. El método de evaluación propuesto debe tener un costo similar a otros métodos existentes, de acuerdo a una métrica que incluye factores de costo relevantes.

Este trabajo propone un método de evaluación llamado Mobile Collaboration Evaluation (MCE). El método consta de tres pasos: (1) el modelamiento del proceso de colaboración móvil que será apoyado por la aplicación, (2) el filtrado de requisitos no aplicables a la situación, y (3) la revisión de la aplicación para chequear si los requisitos han sido considerados o implementados. El método MCE fue probado en tres aplicaciones que apoyan los siguientes escenarios de trabajo: inspección de construcciones, trabajo en hospitales, y emergencias cotidianas de bomberos. Estas pruebas permitieron comprobar si el método de evaluación podía entregar una medida del apoyo a la colaboración que otorga una aplicación, además de ideas para mejorarla. Los resultados son positivos, lo que sugiere que MCE puede ser una herramienta valiosa para ayudar a guiar el desarrollo de una aplicación hacia el apoyo a actividades de colaboración.
Evaluation of mobile shared workspaces to improve their support for collaboration

Evaluation of collaborative systems is a complex task that must not only assess system functionality and usability, but also its support for collaboration. There are several difficulties related to evaluation of these systems, e.g. the complexity of work processes involving several actors and the influence of external factors on collaboration. The introduction of powerful mobile devices and widespread networking has exacerbated this problem. Although there is an opportunity to develop systems for mobile collaboration, few evaluation methods for this situation exist, increasing the uncertainty of project success. Also, aspects of mobile work such as variability of group composition and task flexibility should be taken into account.

This thesis explores the hypothesis that a formative evaluation method based on functional requirements for collaboration support in a specific work context can help measure how well a mobile application supports the collaborative work process. The perspective of this method leaves out other issues such as non-functional requirements and functional requirements not related to collaboration. This evaluation method should have a cost similar to existing evaluation methods, according to a metric introducing relevant evaluation cost factors.

The evaluation method proposed to prove these hypotheses is the Mobile Collaboration Evaluation (MCE) method. The evaluation has three main steps: (1) modeling the collaborative work setting to be supported by the application, (2) filtering inapplicable requirements, and (3) checking the application to see whether it implements the suggested requirements. MCE was tested in three mobile collaborative applications that support the following work scenarios: construction inspections, hospital work and firefighter response to common emergencies. MCE was applied to each system at several phases of the development process, and the obtained results were contrasted with results from other evaluation methods. This allowed us to check whether this evaluation method could give a measure of the collaborative support the application had implemented and ideas to improve it. The results are positive, suggesting MCE can be a valuable tool to help guide development towards supporting collaborative work.
Abstract

Collaborative systems are complex and expensive, and there is no way of predicting how well they will help a group accomplish their joint work. The success of a collaborative system depends on multiple factors, such as the group’s characteristics, the context, and the effects of technology on the collaborative activity it has to support. Evaluation can be summative when it assesses a finished system, or formative when it evaluates a system under development. Formative evaluation is typically more interesting than summative evaluation because it can detect errors and suggest improvements to help guide the development towards a final product that truly supports group work. Otherwise, if formative evaluation is not done, the final product may lack context and be unsuccessful. Nevertheless, in many collaborative system developments, evaluation is conducted in an ad-hoc, informal way or not done at all.

Evaluation is a complex task that must not only assess system functionality and usability, but also its support for collaboration and how it affects the organization. There are several difficulties related to evaluation in this type of system. Some of the main issues are the complexity of work processes involving several actors, the influence of external factors (e.g. personalities, organizational context) on collaboration, and the conflicting perspectives to be considered. Evaluation is also costly, sometimes extending over long periods or requiring users to be interrupted from their work to participate in the evaluation process.
The introduction of powerful mobile devices and widespread networking has exacerbated this problem. Although there is an opportunity to develop systems for mobile collaboration, few evaluation methods for this particular situation exist, increasing the uncertainty of a project’s success. Furthermore, in the case of mobile collaborative systems, evaluation has additional difficulty, since aspects of mobile work such as group composition variability, task flexibility, low interdependence, and unstable connectivity should be taken into account.

This thesis explores the hypothesis that a formative evaluation method, based on functional requirements for collaboration support in a specific work context, can help measure how well a mobile application supports the collaborative work process. The perspective of this method leaves out other important issues such as non-functional requirements (e.g. usability, performance) and functional requirements from the application itself that are not concerned with collaboration. This dissertation also proposes that this evaluation method should have a cost similar to existing evaluation methods, according to a metric introducing relevant evaluation cost factors.

The evaluation method we propose to prove these hypotheses is called the Mobile Collaboration Evaluation (MCE) method. This method is especially designed for software supporting loosely coupled mobile collaborative activities. The evaluation has three main steps: (1) modeling the collaborative work setting to be supported by the application, (2) filtering inapplicable requirements, and (3) checking the application to see whether it implements the suggested requirements.

The first step is done through a formal graph-based model specified using the proposed Mobile Collaboration Modeling (MCM) language. The modeling language describes the relationships among user roles and interaction scenarios that are present in a loosely coupled mobile
collaborative scenario. The generated graphs have several interesting properties, e.g. they can be validated to check if they are coherent, and they can evolve as understanding of a scenario improves.

The second step analyzes the specified interaction scenarios and a pre-defined list of requirements (which was obtained from existing literature and experimental results) to determine which functional requirements, concerning collaboration, must be included in a collaborative application supporting the activity. The obtained list of requirements have to be refined, because some of the proposed requirements could be expensive or complex to include, or may be not applicable in some particular cases.

The third step involves reviewing the product to be evaluated in order to determine if the refined list of requirements is included or not. In case a particular requirement is not included in the current version of the product, the method suggests when it should be included. Therefore, the proposed evaluation method can be considered as formative. Following this strategy, the method helps improve the collaboration support for a particular work scenario.

This thesis also proposes an evaluation framework, i.e., a procedure to organize the whole evaluation process. This framework provides a strategy to combine the evaluation method with other methods evaluating issues not present in MCE, such as usability, efficiency and scalability of the product. Thus, the framework helps provide a comprehensive evaluation.

The MCE evaluation method was tested in three mobile collaborative applications that support the following work scenarios: construction inspections, hospital work and firefighter response to common emergencies. Each application had a development team in charge of building
and evolving the solution in order to provide an appropriate support for the collaborative work. MCE was applied to each system at several phases of the development process, and the obtained results were contrasted with results from other evaluation methods. This allowed us to check whether this evaluation method could give a measure of the collaborative support the application had implemented and ideas to improve it. The results are positive, suggesting MCE can be a valuable tool to help guide development towards supporting collaborative work. The results encourage further research, such as using the evaluation method in new settings and transforming it into a framework to help build collaborative systems. This dissertation contributes to a new way of understanding mobile collaboration scenarios with the goal of building systems that truly support the collaborative processes present in them. It is expected the results of this research will positively impact the development and availability of mobile collaborative applications.
The following papers were published as a result of this research:

**Journal Papers**


- Collaborative Systems: Behind the User Interface, Valeria Herskovic, Sergio F. Ochoa, José A. Pino, Andrés Neyem. Accepted in Journal of Universal Computer Science.


Conference Papers


# Índice general

## Nomenclature

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>xix</td>
</tr>
</tbody>
</table>

## 1. Introduction

1. Mobile Collaborative Work ................................. 2
2. Evaluation ................................................ 3
   1.2.1. Evaluation of collaborative systems .................. 4
   1.2.1.1. Why is evaluation so difficult? .................... 7
   1.2.1.2. Why and how to evaluate ......................... 10
   1.2.1.3. What to evaluate ................................. 11
   1.2.1.4. When to evaluate .............................. 12
4. Problem Description ...................................... 14
   1.3.1. Hypotheses ...................................... 15
   1.3.2. Objectives ..................................... 16
   1.3.3. Solution ....................................... 16
   1.3.4. Methodology ................................... 17
5. Document Structure ....................................... 19

## 2. Mobile Collaborative Work

1. MCW Characterization .................................... 22
2. MCW Requirements ....................................... 27
   2.2.1. Correspondence Matrix ............................. 33
   2.2.2. Grouping General Requirements .................... 35
3. Matching between quadrants and requirements ............... 35
4. Discussion ............................................... 36
## ÍNDICE GENERAL

| 2.4.1. | MCW characterization | 36 |
| 2.4.2. | MSW general requirements | 38 |

### 3. Related Work Analysis

3.1. Evaluation Frameworks | 41
3.2. Evaluation Methods | 42
    3.2.1. Timeline of evaluation methods | 43
    3.2.2. Sample of evaluation methods | 44
        3.2.2.1. Groupware Heuristic Evaluation (GHE) | 46
        3.2.2.2. Groupware Walkthrough (GWA) | 47
        3.2.2.3. Collaboration Usability Analysis (CUA) | 48
        3.2.2.4. Groupware Observational User Testing (GOT) | 48
        3.2.2.5. Human-Performance Models (HPM) | 48
        3.2.2.6. “Quick-and-dirty” Ethnography (QDE) | 49
        3.2.2.7. Performance Analysis (PAN) | 49
        3.2.2.8. Perceived Value (PVA) | 49
        3.2.2.9. Scenario-Based Evaluation (SBE) | 50
        3.2.2.10. Cooperation Scenarios (COS) | 50
        3.2.2.11. Knowledge Management Approach (KMA) | 50
        3.2.2.12. Technology Transition Model (TTM) | 51
    3.3. An initial approach to classifying evaluation methods | 51
        3.3.1. Classification based on stakeholders and product state | 52
        3.3.2. Classification based on type, scope and duration | 53
    3.4. Characterizing evaluation methods | 55
        3.4.1. Evaluation Variables | 55
        3.4.2. Performance Levels | 57
        3.4.3. Evaluation Scenarios | 58
        3.4.4. Classification of methods | 62
    3.5. Evaluation Cost | 63
    3.6. Discussion | 65

xiii
ÍNDICE GENERAL

4. Evaluation Framework ................................................. 69
   4.1. Evaluation Guidelines ........................................... 69
   4.2. Evaluation Lifecycle ........................................... 72
   4.3. Discussion ................................................ 74

5. Mobile Collaboration Evaluation (MCE) Method .................. 76
   5.1. MCE Method ................................................ 76
   5.2. Evaluation Method Characterization .......................... 77
   5.3. Evaluation Method Cost ....................................... 79
   5.4. Discussion ................................................ 80

6. Mobile Collaboration Modeling (MCM) Language ................ 82
   6.1. Related Work ............................................... 82
   6.2. Modeling language definition ................................ 84
       6.2.1. Language Specification .................................. 84
       6.2.2. MCM graph analysis ...................................... 86
           6.2.2.1. Two-node analysis .................................. 87
           6.2.2.2. Three-node cycle analysis ......................... 87
           6.2.2.3. n-node cycle analysis ............................. 93
       6.2.3. MCM Validation ........................................... 95
           6.2.3.1. MCM-V Algorithm ..................................... 95
           6.2.3.2. MCM-V Analysis ..................................... 96
       6.2.4. Modeling language use .................................. 97
   6.3. Software Tool ............................................... 98
   6.4. Discussion ................................................ 100
       6.4.1. MCW Characterization .................................... 101
       6.4.2. MCM Language theoretical evaluation .................. 102

7. Experimental Results ................................................. 104
   7.1. Experimental Method .......................................... 105
   7.2. Urban Search and Rescue ..................................... 105
       7.2.1. Validation Process ....................................... 107
           7.2.1.1. Roles and Communication ........................... 107
           7.2.1.2. Simultaneity ........................................ 107

ÍNDICE GENERAL

7.2.1.3. Reachability ........................................ 109
7.2.2. MCM Graph ........................................... 109
7.2.3. Discussion ............................................ 110
7.3. Construction Inspections ................................. 112
  7.3.1. Description of work .................................. 112
  7.3.2. Software ............................................. 113
  7.3.3. Discussion ........................................... 114
7.4. Healthcare ................................................ 116
  7.4.1. Description of work .................................. 116
  7.4.2. Software Evolution ................................... 117
  7.4.3. MCE Evaluation ....................................... 118
  7.4.4. Discussion ............................................ 120
7.5. Daily emergency management ............................. 122
  7.5.1. Description of work .................................. 122
  7.5.2. MCE Evaluation ....................................... 124
    7.5.2.1. Version 1.0 ...................................... 124
    7.5.2.2. Version 2.0 ...................................... 128
    7.5.2.3. Version 3.0 ...................................... 130
    7.5.2.4. Version 3.1 ...................................... 135
  7.5.3. Focus Group Evaluation ............................... 138
    7.5.3.1. First focus group ................................ 141
    7.5.3.2. Second Focus Group ............................... 144
    7.5.3.3. Third Focus Group ................................. 145
  7.5.4. Discussion ............................................ 146
7.6. MCE Method Cost ......................................... 150
7.7. Discussion ............................................... 152

8. Conclusions and Future Work .............................. 154
  8.1. Summary of research .................................... 154
  8.2. Contributions .......................................... 156
    8.2.1. Evaluation method .................................. 156
    8.2.2. Methodology for organizing the evaluation process .... 156
    8.2.3. Technique for evaluation cost comparison ............ 157
ÍNDICE GENERAL

8.2.4. Software tool to support the evaluation process . . . . . . . 157
8.2.5. Understanding of urban search and rescue scenarios . . . . 158
8.3. Lessons Learned . . . . . . . . . . . . . . . . . . . . . . . . . . . 158
8.4. Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 159
8.5. Future work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 159
  8.5.1. Refine requirements for intermediaries . . . . . . . . . . . . 159
  8.5.2. Usability of modeling language and software tool . . . . . . 160
  8.5.3. Framework for mobile shared workspaces . . . . . . . . . . . 160

A. Further information 161
  A.1. Urban Search and Rescue . . . . . . . . . . . . . . . . . . . . . . 161
    A.1.1. Description of work . . . . . . . . . . . . . . . . . . . . . . . . 163
    A.1.2. Communication in USAR . . . . . . . . . . . . . . . . . . . . . . 166
  A.2. Construction Inspections . . . . . . . . . . . . . . . . . . . . . . . 167
Índice de figuras

1.1. Overview of the evaluation process ........................................... 5
1.2. Almost one third of collaborative systems were not evaluated in a
    formal way [114] ................................................................. 6
1.3. The three steps of the MCE method ........................................... 17

2.1. Work cycle in mobile collaboration ........................................... 22
2.2. Simultaneity and non-simultaneity ......................................... 23
2.3. Classification of collaboration scenarios .................................... 24
2.4. State diagram for a worker with respect to a collaborator .......... 26
2.5. Mobile collaborative system architecture .................................. 28
2.6. $+ x_i$ positively impacts $x_j$, $- x_i$ negatively impacts $x_j$ ...... 34
2.7. Requirements organized according to the collaboration classification 37

3.1. Variables adopted for the Evaluation Framework ....................... 57
3.2. Performance levels, adapted from [126] ................................. 59
3.3. Role-based evaluation ......................................................... 60
3.4. Rule-based scenario ............................................................ 61
3.5. Knowledge-based scenario .................................................... 62
3.6. Classification of evaluation methods ....................................... 63
3.7. Evaluation methods according to their cost ............................. 65

4.1. Summary of guidelines for selection of evaluation methods .......... 70
4.2. Evaluation lifecycle ............................................................. 74

5.1. MCE Framework Variables .................................................... 79
5.2. Cost of MCE compared to other evaluation methods .................. 80
6.1. Types of nodes ................................. 84
6.2. Types of arcs .................................. 85
6.3. Equivalence between (a) 1-m and (b) 1-1 relationship analysis ... 86
6.4. Interaction between two nodes ..................... 87
6.5. Triangle $r_{123}$ .................................. 88
6.6. n-node cycle ................................... 93
6.7. GMT Interface (MAC OSX) ......................... 99
6.8. (a) Graphical display of error in top triangle. (b) Graphical display of error in bottom triangle. .................. 100
6.9. Requirements generation .......................... 101

7.1. Firefighter hierarchy ............................ 106
7.2. Radio communications ............................ 108
7.3. Simultaneity .................................... 108
7.4. Unreachability ................................. 109
7.5. MCM Graph for USAR scenario ................. 111
7.6. ITO interface (a) General view, and (b) detail of the central area [127] .................. 113
7.7. MCM Graph for construction inspections scenario .... 114
7.8. cTracker interface ............................. 117
7.9. MCM Graph for hospital setting .................. 120
7.10. MobileMap v1.0 interface ...................... 125
7.11. File exchange in MobileMap v1.0 ................ 126
7.12. MCM Graph for MobileMap version 1.0 ......... 127
7.13. MobileMap v2.0 interface ...................... 129
7.14. MobileMap v3.0 interface ...................... 131
7.15. Information management in MobileMap ........... 132
7.16. Information deployment and exchange ............ 133
7.17. MCM Graph for MobileMap version 3.0 ......... 136
7.18. MCM Graph for MobileMap version 3.1 .......... 139

A.1. (a) Training grounds (b) Diagram of training grounds ............. 162
A.2. (a) Known building numbers are used (e.g., 701 and 706), unknown ones are filled in. (b) If all numbers are unknown, numbers are assigned for reference. (c) The sides of buildings are numbered. (d) A building is divided into four quadrants. . . . . . . . . . . . . . . . . 164
A.3. Marking system for structures, updated as work progresses. . . . . . . 165
A.4. Radio communication . . . . . . . . . . . . . . . . . . . . . . . . . . . 167
Capítulo 1

Introduction

There are several work settings in which groups of people work together while they are on the move. This type of work is called *mobile collaborative work (MCW)*. Even though these groups of people usually work in a loosely coupled manner, they still need to communicate and interact from time to time. Historically, this has been accomplished in different ways, e.g. through accorded meeting times and places, or through radio communication (such as two-way radio transceivers). In recent years, mobile networking technologies (e.g. 3G, WiFi, WiMAX) and devices (e.g. cellular phones, tablet PCs, PDA’s, netbooks and smartphones) have begun to enable workers to communicate in new ways. However, there is still much that has to be done to effectively support collaborative work while users are on the move.

Many mobile applications are in use today. However, software to support mobile collaboration is still being researched and deployed with varying degrees of success. This thesis presents an evaluation method for collaborative mobile applications. This method has several benefits over existing methods: it helps developers understand the work scenario, it may be used to guide development towards collaborative work support, and it is easy and inexpensive to deploy.
1.1 Mobile Collaborative Work

Software that supports collaborative work is designed to support people interacting with other people. These systems aim to enhance communication, cooperation and collaboration, allowing people to work together to accomplish a common goal [39]. The research area in computer science that studies this type of system is called CSCW, or computer supported cooperative work.

Mobile collaborative work occurs in settings in which individuals are “spatially and/or contextually dispersed and the relevant loci of the individuals change with time during the cooperative act” [28]. This type of work, which occurs in diverse areas such as service technicians, police patrols and firemen [77], as well as hospitals and construction inspections [127], may be supported by mobile telephones or other devices, helping activities to become mobile instead of focused within a particular location [80]. In this case, we may talk about mobile virtual work, which is a setting in which “physically distributed and mobile people interact through digital infrastructures and mobile tools to perform their tasks in an organisational context that has a mobility oriented structure and culture” [3].

Mobility may be critical to some work settings, and it is important to consider this in the design of CSCW applications, so it can be accommodated instead of eradicated [15]. A type of CSCW application to support mobile work is a Mobile Shared Workspace (MSW). A shared workspace is a persistent space with a shared repository that holds the object of group activity, where collaborators interact synchronously or asynchronously [38]. It is important to note that shared workspaces in this definition refers to a class of applications, and not to the data-sharing layer typically present in most collaborative software [52]. Groups may work together through a shared workspace while co-located or distributed, or even while partially distributed, where some members are separated from the core group but linked via computational support [22]. Moving away from the desktop should not hinder the user from accessing information necessary for cooperating with other group members [36]. However, affording mobility to shared workspace users is a challenging task. Development of mobile collaborative systems presents challenges stemming
from the difficulties of developing both mobile and collaborative applications. For example, developers must consider usability (limited screen size and input capabilities), heterogeneity of devices [139], and the possibility of disconnections, among others. At the same time, providing collaborative services such as awareness and context (physical environment, work material, cooperating partners) is crucial. The present work aims to ease the development of MSW and improve their success rate by embedding an evaluation method into their development process.

1.2. Evaluation

Whenever an organization acquires a new information system it is important to conduct an evaluation of the product to make sure it satisfies the needs and expectations of the users and the organization. When a team of developers builds an application, it is also important to measure how well this application supports the users’ work. But what is meant by the word evaluation? A good starting point is the dictionary: evaluation is “to determine the significance, worth, or condition of, usually by careful appraisal and study” ¹.

Gelperin and Hetzel [46] describe the phases of software testing through time. In the early history of software development, testing was equivalent to debugging code. Later on, the concept of testing also incorporated making sure a program solved the problem it was designed for and uncovering errors. The concept of evaluation emerged from software testing and the world of work [125]. Organizations are not only concerned with making sure a system fulfills a certain set of criteria, they also must consider the effects of the computer system on their specific environment.

Evaluation is concerned with ensuring that a system works as desired, adds value to the organization, is better than alternative solutions, or it justifies investments to managers and investors. Evaluation includes the degree in which this

matching is accomplished as well. Evaluation may be *summative*, to assess a finished system, or *formative*, when it is applied to a system that is being developed. This distinction was first proposed in 1967 [136] and has been used not only for collaborative systems, but for traditional software as well as in other areas such as education. Summative evaluation is used to determine the overall quality and impact of the product, while formative evaluation also helps inform redesign and all the stages of development, since usability problems, oversights in design, or other errors may be found during evaluation.

Traditional system evaluation tests both usability and functionality of a system [37]. Some authors subdivide evaluation into several categories, e.g., FURPS [124], which divides quality into five items: Functionality, Usability, Trustworthiness, Performance and Support Capacity. Some of these categories are relatively easy to evaluate through well established metrics, e.g. the average time a webpage takes to load to measure performance. Other issues, such as usability or collaboration support, are not straightforward to measure.

### 1.2.1. Evaluation of collaborative systems

Collaborative systems are complex and expensive, and there is no way of predicting how well they will help users accomplish their joint work. Collaborative systems change the way people communicate, which in turn may change existing work practices. This makes it important to incorporate an adequate support for communication and collaboration into them. Therefore, evaluation of collaborative systems must not only assess system functionality and usability, but also its support for collaboration and how it affects the organization it is inserted into. Otherwise, the development process may conclude with an application that lacks context, does not support the users’ needs for collaboration and is not compatible with the organization.

Haynes et al. [58] cite other factors to explain the importance of conducting evaluation of collaborative systems: first, organizational investment must be justi-
1.2 Evaluation

Figure 1.1: Overview of the evaluation process

Evaluation, i.e., the organization needs to quantify the value of the implemented system. Second, formative evaluation plays a key role in the development cycle, and it should guide the design and implementation of a system. This view of evaluation as central in the development of a system is illustrated in Figure 1.1, which presents a pragmatic overview of the development process based on the star model [64]. In this view, evaluation follows any phase of the development process, requires resources to be carried out, and provides results to interested stakeholders, e.g., developers, managers or investors.

Despite the need for evaluation, many CSCW systems are poorly evaluated or not evaluated at all. A study of 45 articles from 8 years of the CSCW conference revealed that almost one third of the presented collaborative systems were not evaluated in a formal way [114] (Figure 1.2), while a study that also included the ECSCW conference and the Journal of CSCW [120] found few articles that focused on evaluation. A study conducted in 2005 found that the number of papers in computer science including empirical evaluation had not increased significantly in
1.2 Evaluation

ten years [149]. Even when evaluations are conducted, many are ad-hoc, depending on researchers’ interests or appropriateness for a specific environment [70].

Figura 1.2: Almost one third of collaborative systems were not evaluated in a formal way [114]

The reasons for the lack of established evaluation strategies are various. Evaluation, in particular when it requires the participation of multiple users, is expensive and time consuming. Required evaluation resources, such as users, installations, and experts, may be hard to find or unavailable [6]. These expensive methods are not well suited for prototyping and guiding design, since scarce and expensive resources would need to be used frequently. Even if an evaluation is conducted, the effects of collaborative software are not immediate; benefits may be long-term and thus not observable in a laboratory setting [49]. A global perspective of the evaluation results is difficult to obtain, because there are many conflicting perspectives to consider (e.g., stakeholders such as management, designers, and employees, and variables such as usability, communication, and organizational impact). Evaluation methods for single-user systems are not always applicable to collaborative systems, since in these systems, success also depends on the various backgrounds and personalities of group members, social factors, organizational culture and dynamics within the group [49] or the context of the interaction [35]. Several authors have attributed unexpected failures of collaborative systems to designers failing
to incorporate a sophisticated understanding of the users’ workplace that includes the social dynamics [51] and a wide social context consisting of actions, activities, artifacts and work conditions [14, 79]. The next section discusses the difficulties in evaluating collaborative systems in more detail.

1.2.1.1. Why is evaluation so difficult?

The success of a collaborative system depends on multiple factors, including the group’s characteristics and dynamics, the social and organizational context it is inserted in, and the positive and negative effects of technology on the group’s tasks and processes. Therefore, evaluation should attempt to measure several effects on multiple interdependent stakeholders and in various domains. What distinguishes collaborative systems from other information systems is indeed the need to evaluate their impact with an eclectic perspective.

It is desirable to obtain a comprehensive understanding of the effects of a collaborative system. For that purpose, insights of the system’s impact should be obtained in the individual, group and organizational domains. The individual domain considers the positive/negative contributions to the individual goals and tasks. The group domain concerns communication, information sharing, interdependency, mutual adjustment and group efficiency and effectiveness. Finally, the organizational domain encompasses both individuals and groups as perceived as contributing to a major endeavor set up by a common vision, mission and goals.

Ideally, a single evaluation method should cover the individual, group and organizational domains, assessing whether or not the system is successful at the combination of those realms. Unfortunately, no such single method is currently available, if ever will. The fundamental cause for it is related to the granularity and time scale of the information obtained at these three domains [100]:

- The information pertaining to the individual is usually gathered at the cognitive level, focusing on events occurring on a time frame in the order of a few minutes or even seconds.
1.2 Evaluation

- Group information is gathered at the interaction/communication level, addressing activities occurring in the range of several minutes and hours.

- The information regarding organizational impact concerns much longer time frames, usually in the orders of days, months and even years.

Thus, one problem is the adopted measuring instruments must deal with an extremely wide range of observable events and time frames, making it difficult to develop a single evaluation method. Furthermore, the collaboration scenarios are highly variable and thus one of the domains may be emphasized over the others. For instance, in a particular case, the organizational impact of a system may be determined to be the most important evaluation result. This is well known in military aviation, e.g., where organizational objectives overcome usability issues [134], which must be solved with extensive training.

Evaluators must decide which variables should be measured in each domain (individual, group and organization) when an evaluation has to be done. They should select several variables for domains that have been deemed important, and they must also choose the most appropriate time frames for data acquisition. Each variable has a quantitative or qualitative value range, from absence or zero, to full completion or maximum value. Unfortunately, the currently available evaluation methods do not seem to have the flexibility needed to support all these variations and thus several techniques and tools must be used to obtain all evaluation results.

However, the situation has additional complexity. The results of an evaluation should be weighted by the degree of certainty in them, which depends on the development status of what is being evaluated. In the beginning, the product to be evaluated is just a concept or a collection of design ideas, so the results have a high degree of uncertainty. As the development progresses, there may be a prototype or system that can be tested, providing evaluators with an increased degree of certainty and more precise results.
Therefore, a comprehensive evaluation must be an incremental, evolving process to be applied repeatedly throughout the development cycle, from the initial conception of the system, through its analysis, design, implementation, deployment (not necessarily in this order, as presented in Fig. 1.1) and later lifetime. At several development stages, measures should be conducted to assess the system’s current state and to correct deficiencies and improve the design.

Relevant variables should be measured in each of the evaluation phases of a product through appropriate evaluation methods. For example, in the analysis phase, satisfaction may be measured through focus groups, but when the system is functioning, questionnaires may be given instead to real users to measure it. Another variable, which was not relevant in the first stages of development, may be measured in the deployment phase. The results of each evaluation should be used to focus the redesign efforts on problematic areas and to obtain a more balanced evaluation. Certainty in the evaluation results should increase as development progresses from an idea to a finished product.

However, this overall description of the evaluation process is more conceptualized and idealized than practical or even feasible, as it assumes a high level of control over the design and the evaluation processes. In practice, it is highly probable that the design problems, context and ideas change dramatically, so the focus of evaluation may also change in unexpected ways. This intimate relationship between design and evaluation thus makes it even more difficult to develop appropriate evaluation methods for collaborative systems since the methods themselves must evolve in time according to the evaluators’ perceptions and goals.

This view highlights additional causes for the difficulties associated to collaborative systems evaluation:

- **Multiple occurrences**: The evaluation task, positioned at the center of the whole development process, may occur frequently and at any point in time, independently from the current product status.
1.2 Evaluation

- **Multiple forces:** A strict focus on evaluation is difficult to pursue, because there are additional forces to consider within the pragmatic goal of developing a system. For instance, various types of stakeholders (management, designers, shareholders, employees from various areas) may have different impacts on the development process and be in conflict with the evaluation process itself.

- **Cost:** Repeated evaluations, in particular when they require the participation of multiple users, are expensive and time consuming. Also, high cost approaches are not cost effective in the preliminary evaluation stages.

1.2.1.2. Why and how to evaluate

McGrath [86] characterized the purpose of conducting an evaluation into three main goals: precision, generalizability and realism.

The first goal concerns the *precision* of the data obtained by the instrument being used. This goal is inherently linked with the capability to control the dependent and independent variables, the subjects and the experiment. Laboratory experiments are usually selected to accomplish this high level of control.

*Generalizability* concerns the extent the obtained results may be applied to a population. High set goals on generalizability usually imply adopting large-scale inquiries and surveys, while low generalizability is obtained by interviewing a small audience.

*Realism* addresses how closely the obtained results represent real-world conditions, considering the work setting, the population of users, and the tasks, stimulus, time stress, absence of observers, etc. Laboratory experiments have been criticized for providing low realism, especially with collaborative systems, whereas field studies have been considered to score high on realism but low on precision.

Overall, the ideal evaluation should maximize the three goals, for instance using multiple evaluation methods and triangulating the obtained results. Nevertheless, McGrath [86] states this would result in a very costly and difficult to carry out
evaluation, which ultimately may have to be considered utopian. McGrath then identified the major compromising strategies adopted to overcome the costs of an ideal evaluation:

- Field strategies – Set out to make direct observations of realistic work
- Experimental strategies – Based on artificial experimental settings aiming to study specific activities with high precision
- Respondent strategies – Obtaining evidence from sampling a large and representative population
- Theoretical strategies – Using theory to identify the specific variables of interest.

1.2.1.3. What to evaluate

Pinsonneault and Kraemer [119] defined one of the pioneering evaluation frameworks addressing the practical aspects related with what exactly is the object under evaluation. The framework adopts an input-process-output view to conceptualize the relationship between the technology support and other factors related with the group, group behavior and work context:

- Contextual variables – The important factors in the group behavior. Contextual variables belong to five major categories: personal, situational, group structure, task characteristics and technology characteristics (e.g., anonymity and type of communication).
- Group process – The characteristics of the group interaction, including decisional characteristics, communicational characteristics and interpersonal characteristics.
- Outcomes – The outcomes of the group process affected by the technology support, including task-related outcomes and group-related outcomes.
1.2 Evaluation

This framework has been highly influential, especially because it created a common foundation for comparing multiple experiments. Also, the distinction between group process and outcomes highlights two quite different evaluation dimensions commonly found in the literature, the former usually being either more naturalistic (e.g., ethnography [67]) or analytic (e.g., groupware walkthrough [115]), and the latter one being more experimental (e.g., value creation [21]). Other evaluation frameworks, such as the ones proposed by Hollingshead and McGrath [65] and Fjermestad and Hiltz [44], are based on the above framework.

Regarding more recent evaluation frameworks, Neale et al.[99] proposed a simplified evaluation framework consisting of two categories. One encompasses the aforementioned contextual variables. The other category concerns the level of work coupling attained by the workgroup, which combines technology and group process characteristics. Along with this proposition, Neale et al.[99] also recommend blending the different types of evaluation. Araujo et al.[7] also proposed a simplified framework based on four dimensions: group context, system usability, level of collaboration (similar to the level of work coupling), and cultural impact. The cultural impact is seen as influencing the other dimensions, thus introducing a feedback loop in the input-process-output view.

1.2.1.4. When to evaluate

The timing of the evaluation is inherently associated with the development process. It is common to distinguish between the preliminary and final development stages [34, 57]. The preliminary stage affords what has been designated formative evaluation [136], which mainly serves to provide feedback to the designers about the viability of design ideas, usability problems, perceived satisfaction with the technology, possible focal points for innovation and alternative solutions, and also feedback about the development process itself. The final stage, designated summative evaluation, provides complete and definitive information about the developed collaborative system and its impact on the users, the group and the organization.
1.2 Evaluation

1.2.2. Mobile Systems Evaluation

Evaluations of mobile systems to support collaboration, and specifically mobile shared workspaces, have not been extensively studied. Nevertheless, mobile systems are evaluated in other aspects, such as functionality and usability.

Some tools are available to ease the task of choosing appropriate mobile devices to collaborate and build mobile collaboration. Guerrero et al. [53] present a framework that organizes work context elements and relevant device features in order to allow the identification of the most advantageous computing devices to support mobile collaboration in each work context. Some mobile devices are appropriate for some tasks - e.g., for a task involving much mobility (working while walking), PDAs and cellphones are appropriate, while a desktop PC is not due to its size and weight. Alarcón et al. [1] present a framework to design collaborative applications taking contextual characteristics into account. These strategies do not, however, evaluate how well the final applications and devices support collaboration.

When collaboration is carried out through mobile devices, the context of use of the system is continually changing as the user moves. Schmidt et al. [135] propose to characterize two types of mobile context: context related to human factors (such as user, social environment, and task), and context related to the physical environment (conditions, infrastructure, location). In mobile shared workspaces, the human factors need to include the organization, stakeholders, etc. Khalil and Coneley [72] identify four kinds of contextual information relevant to mobile telephony: location, activity, company, and whether users are engaged in a conversation. The importance of context in mobile applications has led some researchers to advocate usability evaluation in the field, while others conduct evaluation in a laboratory. For instance, Kjeldskov and Stage [74] propose usability evaluation techniques that consist of video recording and interviews, while simulating mobility through treadmills and walking through a course in a laboratory setting. Po [121], on the other hand, adds context to heuristic walkthrough, creating a technique called contextual walkthrough. This technique consists of conducting heuristic walkthrough in the real setting of each defined scenario, and provides evaluators a good sense of
the environment, which helps them detect usability problems and those of highest severity. Scenarios are also used to test mobile social software, since they are simple to convey to users, who rapidly comprehend the potential uses of the application and are able to provide suitable feedback to evaluators [23].

1.3. Problem Description

Collaborative systems are complex and expensive, and their development provides several challenges, which must be anticipated and considered for the systems to succeed. Mobile collaboration is inherently flexible because the setting, context, and even user location and availability are constantly changing, so it is difficult for development teams to analyze work scenarios. Traditional analysis methods are not well suited for this task, because they do not consider several relevant aspects of collaborative work. They may, e.g., not deal with collaboration issues or underrepresent the intrinsic variability of group work [118]. Some proposed methods for collaborative systems do not consider several important aspects of mobility such as task flexibility, low interdependence and unstable connections. The design of mobile collaborative systems is underspecified [113] and there is not enough research on non-technical issues (such as the relationship between collaboration and mobility) that are vital to design [4].

Many completed implemented collaborative systems have failed due to various reasons, among them, the difficulty in evaluating these applications [49]. It would be especially important to predict during the development process how well the product will help users to accomplish their collaborative work. Otherwise, the errors and omissions that are encountered when the system is used in its real environment result in changes to the systems which are difficult and expensive to apply on a finished product. A formative evaluation strategy that can be applied throughout the development can help detect errors at an early stage, when fixing them may not be difficult.
The use of mobile devices becomes increasingly widespread every day [32, 123]. However, these devices present challenges to developers because of their limited input and output capabilities and the fact that they can be used from different contexts. Mobile collaborative systems such as mobile shared workspaces share both the challenges of developing mobile applications and constructing collaborative systems that truly support group work. This thesis deals with the problem of guiding the development process of mobile shared workspaces by defining an evaluation method that provides information to developers about how to improve the system for collaboration. This evaluation method is based on the requirements the application has to fulfill, given the collaboration context, to support the collaborative process. The goal is that the development results in a product that truly supports the scenario it was designed for, to improve the possibility of success of the system. This thesis defines the following:

- a formative evaluation method, which is conducted after any step of the development process to measure how well a MSW system supports collaborative work, and
- a framework to organize the evaluation process.

### 1.3.1. Hypotheses

The hypotheses of this thesis are the following ones:

- **H1**: A formative evaluation method based on the functional requirements for collaboration support in a specific work context can help measure how well a mobile shared workspace supports the collaborative work process.

- **H2**: The cost of the proposed evaluation method will be similar to existing evaluation methods, according to the process duration and effort required to conduct it.
1.3.2. Objectives

The main goal of this thesis was to define a formative evaluation method to help predict how well a MSW supports collaborative work. The cost of applying this evaluation method should be lower or equal than the cost of existing evaluation methods.

The specific objectives derived from the main goal are the following ones:

1. **Comparison based on cost.** Define a way to measure the cost of an evaluation method for collaborative systems, to permit the comparison of several evaluation methods.

2. **Capture work scenario.** Propose a method to capture the work scenario of the intended use of an application, establishing the characteristics relevant to supporting collaboration.

3. **Define requirements.** Define a set of requirements that should be present in MSW. These requirements should depend on the collaborative process, the product designed to support it, and the specific context where the system will be used.

4. **Define evaluation method.** Design an evaluation method supported by the defined requirements that permits the prediction of how well the system supports collaboration. The method should also help to solve problems that are obstacles to collaboration.

5. **Create software tool.** Create a software tool to support the evaluation process. This tool should help reduce the cost of the evaluation method.

1.3.3. Solution

The solution was to propose an evaluation method, called mobile collaboration evaluation (MCE) supported by the general requirements for mobile collaborative work in loosely coupled scenarios. The evaluation method consists of three distinct
1.3 Problem Description

steps, illustrated in Fig. 1.3. First, a modeling language called the Mobile Collaboration Modeling (MCM) language is used to create a graph that represents the collaborative process. The modeling language is presented in detail in chapter 6 of this work. Then, the graph is combined with a list of general requirements present in the development of mobile collaborative applications (detailed in chapter 2), to generate a list of suggested requirements for the specific collaborative scenario. Then, the software at its present stage (design, prototype, finished product) is checked to see whether it is implementing the suggested requirements. The result of this revision is the evaluation result, which can be fed back into the software development process to improve the application. The MCE method is explained in chapter 5.

![MCE Method Diagram]

Figura 1.3: The three steps of the MCE method

1.3.4. Methodology

The methodology consisted of six main activities that were carried out in order to study and understand mobile collaboration settings, create an evaluation
method and study the results of its implementation. Some of these activities (for example, the observation of a mobile collaborative setting and the review of mobile collaboration requirements) overlapped in time, which resulted in a richer understanding of both activities. A summary of the sequence of activities is presented below.

**Study of evaluation methods**

The first phase was the study of existing evaluation methods, techniques and frameworks. The main activities were researching existing methods, categorizing them, and talking to researchers about their experiences in collaborative systems evaluation. This phase also provided initial experience in evaluation of collaborative systems, specifically in the evaluation of a collaborative system for electronic brainstorming [43], which provided a practical experience in the challenges and difficulties of collaborative systems evaluation.

**Observation of mobile collaboration**

This phase consisted of the observation of a mobile collaborative work setting. The selected setting was an urban search and rescue training exercise for firefighters after an emergency such as an earthquake. In this scenario, groups of firefighters work in a loosely coupled way to solve a common goal, which is to rescue all the victims as fast as possible. These groups are both autonomous (they have to make quick decisions with the information they have) and under the direction of an incident commander, who directs the global rescue response. The observation of the work resulted in a deeper understanding of how mobile groups work and interact.

**Study of mobile collaboration requirements**

This phase consisted of the study of existing literature to gather a set of mobile collaboration requirements. These requirements were categorized, grouped and described in detail. Afterwards, they were validated in new settings, such as a hospital studied by Mejia et al. [87].
Development of modeling technique for mobile work

The next step consisted of developing a modeling language for mobile collaborative work. This language was based on the loosely coupled mobile workgroups that had been studied. In this step, the language was formally defined to allow automatic analysis and validation. A software tool was designed to aid in modeling mobile collaborative work settings.

Development of evaluation method based on modeling technique

The previous activities - studying mobile collaboration requirements and developing a modeling technique - permitted the creation of an evaluation method for mobile collaboration. The software tool was updated to include evaluation support.

Evaluation of existing applications

Finally, the modeling technique and evaluation method were applied to existing and new applications, to assess the usefulness and applicability of the proposal.

1.4. Document Structure

This document is organized as follows.

- **Chapter 2** presents a description of mobile collaborative work, along with a characterization of the possible interaction scenarios. A review of literature containing requirements for mobile collaborative applications is presented, resulting in a list of general requirements for this type of software.

- **Chapter 3** presents the analysis of related work in relation to collaborative systems evaluation. This chapter categorizes and compares evaluation methods, and proposes a way to compare their costs.

- **Chapter 4** presents the evaluation framework that proposes a way to organize the evaluation process.

- **Chapter 5** presents the MCE evaluation method.
1.4 Document Structure

- **Chapter 6** presents the MCM modeling language, a central part of the MCE method.

- **Chapter 7** presents the experimental results derived from applying the evaluation method in several software development processes.

- **Chapter 8** presents the conclusions of this thesis, as well as the directions of future research.
Capítulo 2

Mobile Collaborative Work

Communication is a driving force behind many of the strategies workers use to interact in mobile collaborative work. Communication not only allows interactions among persons but also resource sharing and the exchange of contextual information. Communication problems in mobile settings strongly decrease the capabilities and opportunities to carry out collaborative interactions, some of which could be urgent or critical. For example, in urban search and rescue, a firefighter may find a group of trapped victims and need to communicate this situation to other companies in order to get help on time. If the communication channel and the people involved are available, the collaborative interactions will be possible and the situation may evolve positively. Otherwise, the consequences will be unpredictable.

In some work scenarios, such as firefighting, most breakdowns in communication are caused by the inexistence, failure or unavailability of a communication channel. When communication channels are more stable and widely available, failures in communication are usually caused by the unavailability of one of the collaborators.

Time is also important when providing support to collaborative processes. Collaborative software is usually organized in a space/time matrix [37]. However, this is not possible in mobile settings, since both time and space may be constantly changing. In these settings, the line between remoteness and co-location becomes blurred, and users may transition between synchronous and asynchronous work.
2.1 MCW Characterization

For this reason, a characterization was proposed and is described in the next section. Then, a survey of the literature led to a list of requirements that are common to all MSW development, which were then organized according to the characterization. This chapter presents the characterization, the list of requirements, and a discussion of their applicability.

2.1. MCW Characterization

Work is usually loosely coupled [29] during mobile collaboration, so workers are autonomous most of the time [41, 95, 101, 102, 113, 116] and carry out sporadic on-demand collaboration processes (Fig. 2.1). After engaging in collaboration, users return to autonomous work. To be able to interact with other users, it is very important that users are able to find the required collaborators and communicate with them. This thesis therefore defines two perspectives, or dimensions, which are of interest to study mobile collaborative work. These dimensions are simultaneity and reachability, which are strongly related to time and communication.

Figura 2.1: Work cycle in mobile collaboration

**Definition 2.1.1.** Reachability is the existence and availability of a communication channel (either physical or virtual), combined with the availability of the two actors involved.
This definition of reachability means that two actors who are reachable will be able to exchange information and communicate in a highly predictable way, i.e., actors can expect a reply within a certain period of time [50]. If an actor is unavailable, or if there is no communication channel between two actors, they are in an unreachable situation, so their communication is fragmented, unpredictable, or impossible.

**Definition 2.1.2.** Simultaneity is the concurrent presence of two actors working towards the common goal, although they may be working in different sub-tasks or in different spaces.

Two simultaneous actors may have the option to participate in a synchronous interaction (which could be co-located or remote). In other words, those interactions in which the participants have active communication during a very short period of time (Figure 2.2). In contrast, in non-simultaneity, one of the actors is absent, so actors must only engage in asynchronous interaction. In these interactions, communication between participants is highly intermittent during a long period of time, which means one (or both) of the participants does not exchange information during a significant period of time.

![Figura 2.2: Simultaneity and non-simultaneity](image)

With these two dimensions, four possible states are created: simultaneous/reachable (SR), simultaneous/unreachable (SU), non-simultaneous/reachable (NR) and non-
simultaneous/unreachable (NU), presented in Figure 2.3. Whenever two actors need to interact, they will be in a particular quadrant of the classification. Next, we briefly describe each of the quadrants.

**Simultaneous – Reachable (SR):**

In this scenario, both actors are working at the same time and each one is able to interact with the other one directly. If both actors so desired, they would be able to communicate and work together in one unbroken interval. Their communication is highly predictable (they have a communication channel and are both available to interact), so they may expect to receive a response in a certain period of time. This type of collaboration scenario is present e.g. when both actors are communicating face-to-face, or when they are working in different places and connected through a wireless network. The SR situation represents a typical mobile

---

**Figura 2.3: Classification of collaboration scenarios**
synchronous collaboration scenario. Several types of technological support for this scenario currently exist, such as Instant Messaging (IM), file transfer, etc.

**Simultaneous – Unreachable (SU):**

In a simultaneous - unreachable scenario, the actors are working synchronously but they are unreachable, and therefore unable to communicate in a predictable way. For example, in a scenario with no network infrastructure, two users may be at work in the same building on related tasks, but in an extended area in which the communication range of the ad-hoc wireless network is not enough to keep them communicated. Or even if there is an available communication infrastructure, one of the collaborators may be unavailable and therefore unreachable to his co-collaborator. The SU scenario is especially sensitive in time-critical work environments, since collaborators may not be able to wait for a natural change from an SU state to a SR or NR state [60].

**Non-simultaneous – Reachable (NR):**

In this scenario, the actors are working in different time periods, but there is an infrastructure that allows them to communicate asynchronously despite their time differences. For example, two users working in different shifts may have an explicit location (server, message board, e-mail address) where they leave notes or messages to make the other aware of problems. Both users are aware of the location, so they know that any problem they report through this infrastructure will be known by the other user. Some technological solutions have been proposed for this scenario, e.g. e-mail and message boards.

**Non-simultaneous – Unreachable (NU):**

In an asynchronous and non-simultaneous scenario, collaboration between two actors is extremely difficult since they are working in different times and lack a way to communicate directly. In addition, there is no permanent intermediary (such
as a server) that allows them to communicate asynchronously. The NU scenario is infrequent, but it may happen e.g., when collaboration is being established or between actors who are unpredictable in their e-mail communications. In this case, work is practically autonomous and very weakly interdependent, however, providing technological support (e.g., mobile devices with network access, a server with which to synchronize work) may ease the collaboration process.

Two actors may pass through several of these states during their work, although not necessarily through them all. Figure 2.4 presents a state diagram for one actor, illustrating the transitions between states that may happen during his workday in relation to a collaborator.

Figura 2.4: State diagram for a worker with respect to a collaborator
2.2. MCW Requirements

There are several general (transversal) requirements that have to be considered when developing any software system. Most of them correspond to quality requirements, such as maintainability, flexibility and reliability [96]. Some specific functional requirements have been described for specific types of systems, e.g., for collaborative systems [83]. We refer to these recurrent functional needs as general requirements. Having a list of general requirements for mobile collaboration could reduce the complexity of the development process. These requirements are usually related to background processes in charge of providing the supporting mechanisms to enable collaboration in a mobile work scenario. Unfortunately, these general requirements are not visible for most users and developers. Therefore, developers could fail to consider these requirements in the development process or they could be included late, jeopardizing the project success.

In order to explain this situation in depth, let us consider the basic architecture of a collaborative system. It is clear that collaboration requires communication and coordination [39]. Requirements related to these concerns should be layered (Figure 2.5). Usually, requirements involved in the upper layer are highly visible to users and developers, because these needs are mainly related to the application front-end. They describe the collaborative system functionalities that a specific application has to expose to end-users. For example, in a workspace supporting students taking notes during lectures on Tablet PCs, these requirements will be related to how annotations are shared and whether other users can edit them or not.

Requirements related to the coordination and communication layers correspond to functionality that is part of the application back-end. Examples of these requirements are: user autonomy, peer dynamic discovery, service and information interoperability, data synchronization, and broadcast messaging. Clients and developers may be unaware of these issues, and thus these requirements become invisible for them. These needs are recurrent in mobile collaborative systems and they mainly involve the communication and coordination layers that are the basis...
2.2 MCW Requirements

Figura 2.5: Mobile collaborative system architecture

for mobile collaboration. Since these general requirements are difficult to identify, no traditional elicitation processes can be used to record them. These requirements frequently produce a gap between the software required by the client and the product’s final version.

This section presents a framework of general requirements that are usually present in the development of mobile collaborative systems [61, 62]. These requirements were identified and organized based on a study of the limitations and challenges of mobile technology, as well as experiences described in the literature and observation of existing mobile collaborative applications. Then, the compiled information is used to present mechanisms to provide the required components to MSW. Each requirement is described below.

Flexibility

Mobile collaborative systems must support frequent changes in group size and structure, as mobility may cause group participants to connect or disconnect from the group [41, 102, 113, 116]. A couple of mechanisms to provide flexibility are the following ones:

- **Automatic peer detection**: The mobile collaborative workspace has to automatically collect and keep information about the reachable peers. In addition, the system has to store the information related to peer availability.
2.2 MCW Requirements

Based on that contextual information the collaborative system could implement awareness mechanisms (i.e. user presence, user availability, or user location) that trigger on-demand collaboration processes.

- **User connection / disconnection:** Applications may allow participants to work offline for most of the time and switch to online use on-demand. Thus, participants will be able to choose their own level of involvement in the collaboration according to their needs and situation. Support may also be included to provide seamless transitions between connected and disconnected phases [81].

**Consistency and Availability**

Frequent disconnections and autonomous work usually cause inconsistencies and unavailability of the resources that are being shared by the group members [41, 101, 130]. Some requirements supporting consistency and availability of data are the following ones:

- **Explicit data replication:** During connection periods, a user should be able to share data with another user so this data will be available to both of them when they are no longer reachable.

- **Caching:** When users collaborate, as much of the shared data as possible should be replicated automatically in each user’s workspace, in order to provide each user with the most up-to-date information when doing subsequent autonomous work.

- **Conflict resolution:** Mobile workers may update local information on the mobile collaborative application when working alone. Eventually, this may generate inconsistencies in the shared data. Data synchronization requires conflict resolution algorithms to reconcile that information in order to have a common view of the shared environment.
2.2 MCW Requirements

Connectivity

Many work scenarios lack a wireless communication infrastructure or the possibility to implement one. In those cases, the application may use a Mobile Ad-hoc Network (MANET) [25, 41, 76, 102, 111, 113, 130, 133]. Connectivity issues should be transparent for the end-user; otherwise, the collaboration capability is at risk. The following mechanisms may be used to provide connectivity to users of a mobile collaborative application:

- **Automatic connection:** The MANET should be automatically formed and kept by collaborative applications running in each participating node. This increases the interaction capabilities among the participants and it increases the availability of the shared resources.

- **Service and device discovery:** Available services and devices (such as public screens, smartboards, file upload, etc) should be automatically detected and seamlessly integrated into the collaborative environment surrounding the physical work context. This may support the collaboration process during casual interactions.

- **Message routing:** This communication mechanism uses intermediary mobile workers to provide reachability between two actors that have more than a one-hop distance between them. Message routing transforms a one-hop network (with a limited communication threshold) into a multi-hop network (with a larger communication threshold) [90].

- **User gossip:** This service is based on gossip sent by a user (interested in starting a collaboration process) to an unreachable partner through the intermediary neighbor nodes. The movement of such nodes (and the unreachable partner) may eventually allow message delivery. The message usually carries information about the requester’s locations in the near future. That information (if it is received by the unreachable partner) could ease communication between them.
Heterogeneity and Interoperability

Regardless of the mobile device used to access the mobile application, a person should be able to interact with any collaborator using the same application [25, 53, 101]. Differences among the devices are a burden to the users that should be eased by the collaborative system. Requirements that should be considered are the following ones:

- **Heterogeneity**: Collaboration may involve heterogeneous devices, such as laptops, PDAs and smartphones. These devices have different hardware features and computing capabilities. Mobile collaborative systems should be designed to support at least message passing among various types of devices.

- **Interoperability**: It is recommended to design a version of a mobile collaborative application for each device being considered for the collaboration process. This ensures the application will take advantage of the particular device features. Data and services interoperability should be ensured in each version of the system in order to prevent the users from becoming unwillingly isolated.

Communication

Mobile collaborative system users need to communicate with each other, and one mechanism to do so is message exchange (e.g., notes, documents or alarms) [25, 55, 102, 113]. Since the users are on the move to carry out their activities, some of them could be unreachable during a time period. Therefore, the system should provide mechanisms for synchronous/asynchronous communication and attended/unattended message delivery. Some of these mechanisms are the ones described below:

- **Synchronous messaging**: When two users are simultaneously available and reachable, they should be able to exchange messages, e.g., to synchronize their shared dataspace or receive notifications. This synchronous communication is the base for synchronous collaboration.
2.2 MCW Requirements

- **Asynchronous messaging:** When two users work at different times, the system should permit them to send messages that will be received when the other user is again available. Examples of asynchronous messaging are electronic mail, or message delivery when a user connects to a server. Like the previous case, asynchronous communication is the base for asynchronous collaboration.

- **File transfer:** Mobile workers carry out weakly interdependent tasks, therefore it is not required for each person to have an instance of each shared resource. Typically members of a particular working group keep, in the local shared workspace, the resources that are relevant to carry out their assigned tasks. New members assigned to that working group could require getting the shared information from their partners in order to start performing a particular task. File transfer is a mechanism required to deal with this need and several other collaborative system services, such as spontaneous collaboration between two users.

- **Pushing notifications:** The messages are delivered to the mobile users at the moment they connect to the server. Typically a pop-up window could be displayed on the user interface to show the pending messages.

**Awareness**

Since a mobile collaborative system supports collaborative work, it must provide awareness mechanisms to improve the users’ understanding of the group’s work [54, 55, 113, 133]. The system should offer offline and online awareness, both of which should be updated as information becomes available. This information should be presented as it occurs, but without overwhelming the recipients, which could diminish their ability to perceive it [110]. Awareness mechanisms are typically used to enhance the collaboration process. The types of awareness that should be supported are at least the following:

- **Online awareness:** Examples of online awareness are lists of connected users, user locations, and current activity.
2.2 MCW Requirements

- **Offline awareness:** Examples of offline awareness are last available modification to a document, and text authorship.

- **Transition awareness:** Awareness about the transitions between connection and disconnection, such as user presence awareness with time memory, or awareness of message delivery.

**Protection**

The collaborative system must incorporate measures ensuring the work of each user is protected [55, 76, 130]. Some of these measures are mentioned below:

- **Ad-hoc work sessions:** The interaction among mobile users should be protected in order to avoid unauthorized participation in the group and invalid access to resources shared among them. Ad-hoc work sessions are a mechanism to deal with this need.

- **User privacy:** Each user should be able to choose which data to share, and some actions may be performed privately. Users will be more likely to collaborate if their privacy is respected. A detailed study of privacy mechanisms for collaborative systems can be found in [106].

- **Security:** The work of each user must be protected so no one can, maliciously or by mistake, destroy someone else’s work.

### 2.2.1. Correspondence Matrix

This section presents a correspondence matrix that illustrates how each general requirement affects the other requirements. The impact may be positive (if it contributes to the other’s accomplishment), negative (the contrary case) or neutral (both requirements are independent). Figure 2.6 shows the following correspondences between requirements:
In the matrix, we may observe the following correspondences:

- **Flexibility**, or the fact that the group may become disconnected at any moment, negatively impacts consistency, connectivity, communication and awareness. The cause for this association is that frequent disconnections - loosely coupled work - decrease group cohesion and the possibilities for direct or indirect communication.

- Consistency positively impacts awareness, since shared information is replicated every time users communicate. At any moment, replicated information may be used for offline awareness of other users’ work.

- Heterogeneity is negatively related to communication because the more heterogeneous a group is in term of devices and software, the more difficult it is to build mobile collaborative systems to support the group’s communications.

- An increase in awareness information provided to users causes their privacy to decrease.
2.2.2. Grouping General Requirements

Requirements may be grouped as they often appear in mobile collaborative application development. Some requirements are tightly related to others, so that when one requirement is identified for a development project, it is highly probable that its related requirements are also present.

*Flexibility, Consistency:* When users work autonomously, flexibility is maximized, since users are usually disconnected and they only connect for short periods of time. In this case, the periods of connectivity must be used to replicate and copy as much information as possible, to be available to users later on when they are disconnected. This combination of requirements is usual in loosely coupled group work.

*Connectivity, Awareness, Protection:* The automatic detection and configuration of a network of users and devices requires awareness mechanisms to ensure users understand who is in the network and how they might interact. Automatic connectivity also requires protection from malicious users trying to connect to the network and privacy mechanisms in case, e.g., a user does not want to be contacted.

2.3. Matching between quadrants and requirements

Requirements may also be grouped according to the collaboration scenario quadrants in which they are required. Since the relationship between two users may transition from one quadrant to an adjacent one, requirements may also be needed to ease the transitions. This grouping permits developers to know which requirements they should consider during system development according to how collaboration will take place. Each requirement may be present in one or more collaboration scenarios or transitions. Some requirements (service and device discovery, heterogeneity, interoperability, user privacy, and security) have not been
added because they are not related to a collaboration situation between two users.

The most intensive collaboration scenario occurs when two users are working synchronously and are reachable. This situation requires them to be able to work together collaboratively by exchanging messages, files and data, as well as providing services that improve collaboration such as awareness and automatic connectivity. On the opposite extreme is the non-simultaneous and unreachable situation, in which collaboration between two users cannot take place because there is no way for them to interact. In this case, the only suggested requirement is to provide offline awareness by displaying awareness information received during connection periods.

Figure 2.7 presents the requirements that are present in each of the collaboration scenarios. Each quadrant also displays a square that represents the category each requirement belongs to. From this figure, we may observe, e.g., that some type of awareness must always be present and that efforts should be made to connect two users even if they are neither reachable nor working simultaneously.

2.4. Discussion

2.4.1. MCW characterization

The dimensions chosen to characterize mobile collaborative work were reachability and simultaneity, over the original classification of collaborative systems using place and time. Naturally, other perspectives could have been used instead. However, reachability and simultaneity are very relevant and can be applied and used to analyze a wide variety of types of mobile work. After using these dimensions in several settings, we can observe the following:

- The dimensions of reachability and simultaneity seem to capture the flexibility of mobile work. In this type of work, workers are dispersed in time and space, and both time and space may be continuously changing. For example,
Figura 2.7: Requirements organized according to the collaboration classification
in emergency situations, firefighters work concurrently in the affected area to solve a common problem. However, since the area may be large and work requires them to move around it, their physical space changes. In hospitals and home care treatment, work is done in shifts, so two persons may be working at the same time in one moment and at different times in the next moment. These are examples of the flexibility that may be captured in the way mobile work is done.

- People are becoming increasingly reachable (e.g. through 3G, WiMax, and WiFi hotspots). However, the wide availability of online connections does not mean they are always reliable. Connections may not exist, e.g. in airplanes, or after an earthquake damages the infrastructure, or when networks become saturated. Even if we could imagine a future where these problems would be overcome and technology would allow everyone to be online always with a rich and stable communication channel, we can be certain users at times will be unavailable. In fact, we can assume some users will purposefully become unreachable and not respond collaboration requests in a predictable way.

### 2.4.2. MSW general requirements

The requirements proposed in this section are based on extensive work done by many researchers and our experience developing mobile collaborative systems. Beginning development with such a list would greatly ease building the software, especially for developers without experience in this area, since these requirements are usually hidden and they should be considered in each development. The presented requirements are summarized in Table 2.1. Naturally, these requirements are a fraction of all the mobile collaboration back-end requirements. Other requirements should also be considered, e.g., requirements to support the autonomous work that users do in periods of isolated work.

Ignoring the presented requirements may cause developers to build a system from scratch that is missing a fundamental component that supports or eases collaboration. Of course, some requirements may be intentionally ignored according to the particular requirements for the application being developed. For instance,
Cuadro 2.1: General requirements for collaboration between two users in MCW

| Flexibility | Automatic Peer Detection | Software must include information on available peers when they are reachable. |
| User connection and disconnection | Each user can choose whether to be on-line or not. |
| Consistency and Availability | Explicit Data Replication | When reachable, users must be able to send and receive information that they explicitly choose. |
| | Caching | When reachable, the workspace should use a background process to replicate as much information as possible. |
| | Conflict Resolution | There must be a conflict resolution strategy to consolidate the workspace if it is in an inconsistent state after two users have worked autonomously. |
| Connectivity | Automatic Connection | Connections to other users and devices should be automatic, without requiring user input. |
| | Service and Device Discovery | Available services and devices should be automatically integrated. |
| | User gossip | The software must permit message delivery for unreachable roles through intermediary nodes. |
| | Message routing | The system must provide reachability through intermediary mobile workers. |
| Heterogeneity and Interoperability | Heterogeneity | Support for heterogeneous devices. |
| | Interoperability | A version of the application should be designed for each device. |
| Communication | Synchronous Messaging | Users should be able to send messages (or notices, warnings, etc) synchronously. |
| | File Transfer | Users should be able to transfer files and other resources to other users. |
| | Asynchronous Messaging | Users should be able to send messages asynchronously, which the recipient will receive when he is available. |
| | Pushing Notifications | Messages may be delivered to a mobile user at the moment he connects to the server. |
| Awareness | Online Awareness | The system must provide appropriate online awareness mechanisms. |
| | Transition Awareness | The system must provide awareness of the transitions between states of other users. |
| | Offline Awareness | The system must provide appropriate offline awareness mechanisms. |
| Protection | Ad-hoc work sessions | The system must provide mechanisms to secure work sessions between users. |
| | User privacy | Each user may choose which data to share. |
| | Security | The work of each user must be protected from other users. |
a mobile application whose goal is to have as many users as possible reading and commenting news articles will probably not need to implement ad-hoc work sessions to protect shared data.

Classifying the requirements for mobile collaborative systems according to the possible collaborative scenarios allows us to realize that not all mobile collaboration takes place in the same way. Therefore, the requirements that should be present in an application depend on the types of interactions users need.
Capítulo 3

Related Work Analysis

This chapter presents a review of collaborative system evaluation that discusses existing methods and frameworks. We define the following terms for the analysis:

- **Evaluation Tool**: An evaluation tool is a single instrument intended to measure system variables. For example, a questionnaire is an evaluation tool that allows evaluators to obtain user opinions.

- **Evaluation Method**: An evaluation method is a procedure used to apply evaluation tools with a specific goal. For example, the Perceived Value evaluation method [5] uses evaluation tools such as questionnaires and checklists with the goal of determining the organizational impact of meetingware.

- **Evaluation Framework**: An evaluation framework is a macro strategy used to organize the evaluation process. Several evaluation methods and tools may be included in an evaluation framework.

This chapter is organized as follows. First, existing evaluation frameworks are discussed. Then, a set of relevant evaluation methods and their characteristics are presented. Next, these methods are analyzed and a way to select an appropriate method for a particular scenario is proposed. Then, we present a proposal to characterize evaluation methods according to several variables and performance levels. The final section introduces a way to compare evaluation methods according to their costs, which was one of the goals of this thesis.
3.1. Evaluation Frameworks

This section presents existing evaluation frameworks and strategies to organize the evaluation process. Several evaluation frameworks propose strategies to include evaluation in the software development cycle [9, 64, 66, 68], while others identify dimensions of collaborative systems to consider when conducting evaluation [7, 33, 66, 119, 129].

The star model proposes evaluation as the central phase in the software development cycle [64]. This means evaluation is conducted after every step of the development. In [9], development is an iterative process of design, implementation and evaluation. Appropriate evaluation techniques are proposed after each development phase. The concept design is evaluated through interviews, the functionality and interface design through a usability test, the prototype through heuristics, the deliverable system through usability tests and finally, the system enhancement and evolution is evaluated through interviews, questionnaires and analysis.

In the lifecycle-based approach [66], an evaluation plan is defined before starting development. Then, after analysis, evaluation instruments are designed, implemented and improved through user feedback. Evaluation is conducted within five domains: context, content, process, stakeholders and success factors. The E-MAGINE framework [68], has a similar structure: first, a meeting and an interview are conducted to establish the evaluation goals and group profile. This information guides the selection of evaluation methods and instruments that are used.

Damianos, Hirschman et al. [33] present a framework based on Pinsonneault and Kramer’s proposal [119]. The framework has four levels: requirement, capability, service and technology. Appropriate measures are selected at each level to conduct evaluation. At the requirement level, measures are e.g. overall system performance, quality, and efficiency. At the capability level, evaluation is based on how well system capabilities support group requirements. At the service level, evaluation is based on comparisons of performance and cost. Finally, at the technology
level, benchmarks such as bandwidth are compared among specific implementations.

The PETRA framework combines the perspective of the evaluator and the perspective of the users, or participants [129]. In this way, it aims to achieve a balance between theoretical and practical evaluation methods.

The CSCW Lab proposes four dimensions to consider when evaluating collaborative systems: group context, usability, collaboration and cultural impact [7]. Each dimension is a step of the evaluation process, which consists of characterizing the group and work context, measuring usability strengths and weaknesses and collaboration capabilities, and studying the impact of the application over time.

3.2. Evaluation Methods

In this section, we present an overview of existing evaluation methods. First, we illustrate the evolution of collaborative system evaluation through a timeline that lists relevant articles and studies. Then, we briefly describe several representative evaluation methods and compare their main characteristics.

An interesting study that complements this characterization is presented in [114]. This survey categorizes the evaluation methods used in eight years of the CSCW conference, and uses some of the same categories used here. For example, this paper shows that 56% of conducted evaluations were formative.

3.2.1. Timeline of evaluation methods

In Table 3.1 we provide a timeline showing the emergence and life of several evaluation methods through time. The preliminary analysis of this timeline showed there are no major trends or stages in collaborative systems evaluation. Most methods seem to have been adopted and used in an ad-hoc manner. For instance,
3.2 Evaluation Methods

Some techniques seem to be revisited from time to time, such as task analysis, walkthroughs and usability studies. Nevertheless, some patterns seem to emerge:

- The adaptation of single-user evaluation methods, developed in the Human-Computer Interaction field, to the specific context of collaborative systems. This has occurred, for instance, with walkthroughs (structured walkthroughs, cognitive walkthroughs, groupware walkthroughs), heuristic evaluation (heuristic evaluation, heuristic evaluation based on the mechanics of collaboration) and scenario-based evaluation.

- The assimilation of perspectives, methods and techniques from other fields beyond technology development. The clearest example is ethnography (observational studies, quick-and-dirty ethnography, workplace studies), but cognitive sciences also seem to have impact (KLM, cognitive walkthroughs, computational GOMS).

- The increasing complexity of the evaluation context. Most early methods (e.g., structured walkthroughs, KLM, discount methods) seem to focus on very specific variables measured under controlled conditions, while some of the later methods seem to consider more broad contextual issues (e.g., multifaceted evaluation, perceived value, evaluating collaboration in co-located environments, lifecycle-based approach).

3.2.2. Sample of evaluation methods

This section presents a sample of evaluation methods for collaborative systems [63]. Each one is briefly described to give an overview of it and the steps that must be followed to apply it. These methods can be directly used to evaluate a collaborative system, or they can be part of an evaluation framework. The evaluation methods in this survey are those directly applicable to collaborative systems.

Table 3.2 presents a summarized characterization of the selected evaluation methods, describing the purpose of the evaluation (why), the evaluation tools that are used in each method (how), the outcomes of the evaluation (what), and the
3.2 Evaluation Methods

Cuadro 3.1: Timeline of evaluation methods

<table>
<thead>
<tr>
<th>Year</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1978</td>
<td>Structured walkthroughs</td>
</tr>
<tr>
<td>1980</td>
<td>Keystroke-Level Model (KLM)</td>
</tr>
<tr>
<td>1989</td>
<td>Discount usability engineering</td>
</tr>
<tr>
<td>1990</td>
<td>Heuristic evaluation</td>
</tr>
<tr>
<td>1991</td>
<td>Observational studies</td>
</tr>
<tr>
<td>1992</td>
<td>Interface walkthroughs</td>
</tr>
<tr>
<td>1993</td>
<td>Cognitive walkthroughs</td>
</tr>
<tr>
<td>1994</td>
<td>Observational studies</td>
</tr>
<tr>
<td>1995</td>
<td>PETRA: Participatory evaluation through redesign and analysis</td>
</tr>
<tr>
<td>1996</td>
<td>Usability studies</td>
</tr>
<tr>
<td>1997</td>
<td>Groupware task analysis</td>
</tr>
<tr>
<td>1998</td>
<td>Formal evaluation of collaborative work</td>
</tr>
<tr>
<td>1999</td>
<td>Multi-faceted evaluation for complex, distributed activities</td>
</tr>
<tr>
<td>2000</td>
<td>Mechanics of collaboration</td>
</tr>
<tr>
<td>2001</td>
<td>Usage evaluation</td>
</tr>
<tr>
<td>2002</td>
<td>Human performance models</td>
</tr>
<tr>
<td>2003</td>
<td>Collaboration usability analysis</td>
</tr>
<tr>
<td>2004</td>
<td>Scenario-based evaluation</td>
</tr>
<tr>
<td>2005</td>
<td>Knowledge management approach</td>
</tr>
<tr>
<td>2006</td>
<td>Performance analysis</td>
</tr>
<tr>
<td>2008</td>
<td>Tabletop collaboration usability analysis</td>
</tr>
</tbody>
</table>
moment in which evaluation is conducted (when). This categorization does not include the popularity (e.g. number of uses) of each method, as not every use is reported, even though it is possible to venture some general-purpose methods (e.g. GOT) have been more intensively used over the years than others.

Cuadro 3.2: Characterization of evaluation methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Why</th>
<th>How</th>
<th>What</th>
<th>When</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHE</td>
<td>Precision</td>
<td>Software analysis, checklist</td>
<td>Effectiveness, efficiency, satisfaction</td>
<td>Summative</td>
</tr>
<tr>
<td>GWA</td>
<td>Precision</td>
<td>Software analysis</td>
<td>Effectiveness, efficiency, satisfaction</td>
<td>Formative</td>
</tr>
<tr>
<td>CUA</td>
<td>Precision</td>
<td>Software analysis</td>
<td>Effectiveness, efficiency, satisfaction</td>
<td>Formative</td>
</tr>
<tr>
<td>GOT</td>
<td>Realism</td>
<td>Observation, checklist</td>
<td>Effectiveness, efficiency, satisfaction</td>
<td>Summative</td>
</tr>
<tr>
<td>HPM</td>
<td>Precision</td>
<td>Interaction analysis</td>
<td>Group performance</td>
<td>Formative</td>
</tr>
<tr>
<td>QDE</td>
<td>Realism</td>
<td>Observation</td>
<td>Redesign</td>
<td>Summative</td>
</tr>
<tr>
<td>PAN</td>
<td>Generalizability</td>
<td>Formal analysis</td>
<td>Efficiency</td>
<td>Formative</td>
</tr>
<tr>
<td>PVA</td>
<td>Realism</td>
<td>Questionnaire, checklist</td>
<td>Organizational impact</td>
<td>Formative</td>
</tr>
<tr>
<td>SBE</td>
<td>Realism, Precision</td>
<td>Interviews</td>
<td>Organizational contributions</td>
<td>Formative</td>
</tr>
<tr>
<td>COS</td>
<td>Realism</td>
<td>Interviews, observation</td>
<td>Redesign</td>
<td>Formative</td>
</tr>
<tr>
<td>KMA</td>
<td>Generalizability</td>
<td>Software analysis, checklist</td>
<td>Knowledge circulation</td>
<td>Formative</td>
</tr>
<tr>
<td>TTM</td>
<td>Generalizability</td>
<td>Interviews, observation</td>
<td>Predicted actual use</td>
<td>Formative</td>
</tr>
</tbody>
</table>

### 3.2.2.1. Groupware Heuristic Evaluation (GHE)

This method is the adaptation of the Heuristic Evaluation (HE) method, in which a group of usability experts visually inspect an interface and judge its compliance with a set of usability principles called heuristics. GHE adapts HE to
collaborative systems [13].

GHE is based on eight groupware heuristics defined by Baker et al. [13] and inspired by the mechanics of collaboration, the fundamental types of collaborative actions defined by Gutwin and Greenberg [55]. These heuristics act as a guideline, or checklist of characteristics a collaborative system should have. Evaluators who are experts in the heuristics qualitatively examine the interface and record each problem they encounter, the violated heuristic, a severity rating and optionally, a solution to the problem. These reported problems are then filtered, classified according to the violated heuristic, and consolidated into a final list, which is used to improve the application and correct its errors during or after the system development.

3.2.2.2. Groupware Walkthrough (GWA)

GWA is a usability evaluation method based on cognitive walkthrough (CW) [115]. In CW, evaluators step through task sequences to explore the actions users will perform. Using CW in a multiple-user setting is complex [40] due to the need to specify group tasks and manage multiple users and task descriptions concurrently.

In GWA, a scenario is a description of an activity that includes the users, their knowledge, the intended outcome, and circumstances surrounding it. In order to build scenarios, evaluators observe users at work and identify episodes of collaboration. Evaluators walk through the tasks in a laboratory setting to determine how collaboration between group members is supported. Each evaluator, taking the role of all users or one in particular, walks through the tasks, recording each problem he encounters. A meeting is then conducted to analyze the results of the evaluation.
3.2 Evaluation Methods

3.2.2.3. Collaboration Usability Analysis (CUA)

CUA is a task analysis technique focused on the teamwork aspects of collaboration in shared tasks [118]. 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 proposes that each collaborative action can be mapped to a set of collaboration mechanisms, or fine grain representations of basic collaborative actions [48]. These collaboration mechanisms can be related to elements in the user interface. The resulting diagrams capture three types of information: details about task components, a notion of the flow through them and how tasks are distributed among group members.

3.2.2.4. Groupware Observational User Testing (GOT)

GOT is based on the observational user testing method (OUT), which involves evaluators observing how users perform particular tasks supported by a system in a laboratory setting [49]. 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. The GOT technique follows the same principle, but evaluators focus on collaboration aspects and analyze users’ work through predefined criteria such as the mechanics of collaboration.

3.2.2.5. Human-Performance Models (HPM)

Human-performance models, e.g. the keystroke level model (KLM), describe the way a person interacts with a physical interface at a low level of detail based on a cognitive architecture. The HPM method [6] adapts this model to evaluate a group of users communicating through a shared workspace. First, evaluators decompose the physical interface into several shared workspaces, and define the supported communication and coupling mechanisms. Second, they define critical scenarios focused on the collaborative actions for the various shared workspaces. Finally, evaluators compare group performance in the critical scenarios to predict
3.2 Evaluation Methods

Execution times.

3.2.2.6. “Quick-and-dirty” Ethnography (QDE)

In QDE [67], evaluators do brief ethnographic workplace studies to provide a general sense of the setting for designers. QDE accepts the impossibility of gathering a complete, detailed 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.

3.2.2.7. Performance Analysis (PAN)

In PAN [11], the application to be studied is modeled as a task to be performed by a number of people in a number of stages, and the concepts of result quality, time, and total amount of work done are defined. 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.

3.2.2.8. Perceived Value (PVA)

PVA measures the perceived value attributed to a meetingware system by its users [5]. This method begins by developers identifying relevant components for system evaluation. Then, users and developers negotiate the relevant system attributes to be evaluated by users. After the users have worked with the system, they fill out an evaluation map by noting whether the components support the attributes or not. Using these ratings, a metric that represents the PV is calculated using a formula including the relevant components, concrete attributes, and ratings made by users.
3.2 Evaluation Methods

3.2.2.9. Scenario-Based Evaluation (SBE)

In a field evaluation using SBE [58], evaluators perform semi-structured interviews to users to discover scenarios, or detailed descriptions of activities, and claims about them. The interviews are coded to identify fragments corresponding to scenario description, claims, requested features, etc. 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.

3.2.2.10. Cooperation Scenarios (COS)

In COS [141], evaluators conduct field studies, semi-structured interviews, and workplace visits. Through these activities, they identify scenarios, cooperative behavior, users involved in it, their roles and the relevant context. Cooperation scenarios are discussed and validated between evaluators and users to achieve maximum fidelity with actual work practices. For each role involved in the cooperative activity, evaluators analyze the new design to see how the task changes and who benefits from the new technology. Then, practical evaluation of system design is performed in a workshop with actual users, where the prototype is presented to end users by means of a cooperation scenario, and design flaws are discovered.

3.2.2.11. Knowledge Management Approach (KMA)

Evaluation using KMA [148] measures whether the system helps users detect knowledge flows and disseminate, store and reuse knowledge. The knowledge circulation process is comprised of six phases: knowledge creation, accumulation, sharing, utilization, internalization and integration. These are also the areas to be evaluated by this approach. To perform evaluation, each area has a list of associated questions, which may be used as a checklist by evaluators.
3.2.2.12. Technology Transition Model (TTM)

TTM [20] predicts the actual system use as a function of the intent to use the system, the value that users attribute to it, how frequently it will be used and the perceived cost of transition. This model proposes that users weigh all factors affecting the perceived value of a system, producing an overall value that corresponds to their perception of the usefulness of the system. To evaluate the predicted actual use of a system, users’ opinions are obtained by interviews, archive analysis and observations. In this way, a collaborative system can be evaluated to increase the speed of its acceptance, while reducing the risk of technology transition.

3.3. An initial approach to classifying evaluation methods

Whenever a stakeholder needs to choose an evaluation method, he/she does it for a specific context. For instance, a project manager may want to determine how well the functionality of a system under development matches the expectations of an organization. The context drives the manager to consider some key features of each method in order to establish which ones could be appropriate.

Considering several features in the selection process will make the list of potential methods short and accurate. If the suitable methods list is empty, then an ad-hoc evaluation method should be designed. If the list contains more than one method, then the evaluator can choose one based on a prioritization of their key features. We call these key features of a method its classification criteria. The next sections present two classifications of evaluation methods.
3.3.1. Classification based on stakeholders and product state

Table 3.3 presents a classification that considers the concerned stakeholders (developers, users, or the organization) and the state of the product (under development or finished). A brief explanation of each category is included.

**Evaluation methods for systems under development**

While a collaborative system is under construction, the developers require formative and inexpensive evaluation methods that allow them to test the product, discover its flaws, and redesign it accordingly. These methods are usually done in a laboratory setting without users. The users of this system could be interested in ensuring that the system works as desired and allows effective and efficient collaboration. Evaluation methods must thus involve users and focus on their opinions. Finally, the organization as a whole requires that a collaborative system improves work, efficiency and the quality of results, allowing managers to justify investments in the technology.

**Evaluation methods for finished products**

Organizations acquiring a collaborative system may require developers to adapt the product to their needs. Also, the development team may need to conduct a post-mortem analysis of a finished product. Evaluation methods should be summative and measure the matching between product functionality and the users and organizational needs, permitting developers to improve the system. The users of the system need to ensure the system works as desired and allows effective and efficient collaboration. Similar to the previous case, methods must involve users and focus on their opinions; however, in this case methods may be summative. Finally, an organization acquiring a collaborative system must go through the adoption of the technology, so methods should be summative, tested in the real work setting,
3.3 An initial approach to classifying evaluation methods

Cuadro 3.3: Method categorization based on stakeholders and product state

<table>
<thead>
<tr>
<th></th>
<th>Developers</th>
<th>Users</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Products Under</strong></td>
<td>GHE, GWA, CUA,</td>
<td>SBE, COS, EMA,</td>
<td>PVA</td>
</tr>
<tr>
<td><strong>Development</strong></td>
<td>HPM, PAN</td>
<td>KMA</td>
<td></td>
</tr>
<tr>
<td><strong>Finished Products</strong></td>
<td>GOT, QDE</td>
<td>GOT, QDE, PAN,</td>
<td>PVA, TTM</td>
</tr>
</tbody>
</table>

and measure how well a system fits in the organization.

It is possible to identify an initial set of relevant methods for a particular scenario considering these criteria. Stakeholders’ identification and product state can be used as initial evaluation criteria because they are fast to instantiate and highly relevant.

3.3.2. Classification based on type, scope and duration

Table 3.4 classifies the evaluation methods considering the people’s participation, time of application, evaluation type, place, time span and goal. This table should be used in the same way as Table 3.3 in order to perform the selection process. The classification criteria included in Table 3.4 are briefly described below.

The first criterion describes *who* participates in the evaluation besides evaluators: users (U), developers (D), experts (E), or combinations of them. This criterion helps determine the viability of a method based on human resource availability.

The *time* to apply the method may be: before the system is designed to test its feasibility (B); during the development process to identify redesign needs (D); or when the application is finished (F) as a summative evaluation. This criterion
3.3 An initial approach to classifying evaluation methods

Cuadro 3.4: Classification of evaluation methods

<table>
<thead>
<tr>
<th>Eval. Method</th>
<th>Who</th>
<th>When</th>
<th>Type</th>
<th>Loc.</th>
<th>Time</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U D E</td>
<td>B S F</td>
<td>N Q</td>
<td>W L</td>
<td>H Y K</td>
<td>P C X</td>
</tr>
<tr>
<td>GHE</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>GWA</td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>CUA</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>GOT</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>HPM</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>QDE</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>PAN</td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>PVA</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>SBE</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>COS</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>EMA</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>KMA</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>TTM</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

helps in the selection depending on the level of progress of the project.

The evaluation *type* establishes whether the collected data is quantitative (N) or qualitative (Q). Quantitative data is useful to compare the results of several evaluations, while qualitative data usually consists of human judgment and may be used for the most complex situations.

The evaluation *place* determines the location where evaluation is carried out, either a laboratory setting (L) or the users’ actual workplace (W). Based on place availability, it is possible to determine whether a certain method may be used or not.

The *time span* of each method goes from hours (H) to days (Y) or weeks (K). This must be considered to establish whether there is enough time to do an evaluation.

The evaluation *goal* describes the objective of the evaluation, which can be to evaluate the product functionality (P), the collaboration process supported by the
3.4 Characterizing evaluation methods

This section proposes a way to characterize evaluation methods according to several relevant variables. We first present these variables and discuss the reasons they were chosen. Then, we present a three-layered model of human performance proposed by Reason [126]. This model, combined with the variables, can be used to describe evaluation methods. We finally organize existing evaluation methods into this classification.

3.4.1. Evaluation Variables

Chapter 2 introduced the need to choose variables to assess a collaborative system under development. We can use the same approach to characterize the evaluation to be done at each evaluation step. We should, then, choose variables providing information on the methods to be applied.

A starting point are McGrath’s evaluation goals: precision, generalizability and realism. These goals are fundamental to layout the evaluation methodology. For our evaluation framework, it seems thus appropriate to choose variables associated to these goals; if the evaluation methods change in succeeding evaluations, these variables will reflect the new evaluation methodology (Figure 3.1). Precision, generalization and realism are then the first three variables used to describe the evaluation method.

It is important to incorporate the level of system detail (depth) as one of the dimensions to characterize the evaluation activities. Notice this variable is quite distinct from the previous three ones. In particular, realism refers to whether the evaluation will use real settings or not; detail concerns the granularity of the evaluation. Precision, on the other hand, focuses on the accuracy of the measuring
3.4 Characterizing evaluation methods

tools.

Another dimension we would like to incorporate in the evaluation framework is the scope (breadth) of the system being evaluated. An evaluation having a large value for this variable would mean the system being evaluated has many functionalities and components being assessed. This variable complements the detail. While the first three variables in this framework consider theoretical issues, the system detail and scope concern the design methodology.

Finally, the invested time variable describes the time used by the evaluators to carry out the work. This variable may not be completely independent from other variables, e.g. detail and scope (since, e.g., a coarse-grain evaluation narrowing to a few functionalities will probably require less invested time). However, from a more practical than theoretical standpoint, it is an important variable to distinguish the efficient evaluation methods from those which are not.

Other variables could be added, or they may be proposed as candidate variables to replace one of the previous six mentioned above. Nevertheless, the variables presented in Figure 3.1 seem to be adequate to analyze evaluation methods, as shown in succeeding sections.

Figure 3.1 shows a radar-graph representation of the evaluation variables: a specific method is represented by a dot in each of the axes (variables). Each axis has a scale from 0 (or minimum value) in the origin to a certain maximum value. These dots may be joined to show a certain evaluation shape. It may be noticed a numeric value for the area within a shape does not make much sense, since the scales are not the same for each variable. However, a light evaluation procedure will probably have low values for most or all variables, whereas a heavy one will probably score high in several evaluation variables.
### 3.4.2. Performance Levels

Reason [126] proposed a three-layered model of human performance in organizational contexts. We will apply this model to the specific context of collaborative systems evaluation.

The model categorizes human performance according to two dimensions: situation and situation control. According to the situation dimension, the organizational activities may be classified as: (1) *routine*, when the activities are well known by the performers and accomplished in an almost unconscious way; (2) *planned*, when the activities have been previously analyzed by the organization and thus there are available plans and procedures to guide the performers accomplishing the intended goals; and (3) *novel*, when the way to achieve the intended goals is unknown to the organization and thus human performance must include problem analysis and decision-making activities. Shared workspaces, workflow systems and group support systems are good examples of collaborative systems technology supporting the routine, planned and novel dimensions.

The other dimension concerns the level of control the performers may exert
while accomplishing the set goals. The control may be *mechanical*, when human action is performed according to a predefined sequence imposed by the technology. The control may be *human*, when the technology does not impose any predefined action sequence. And finally the control may be *mixed*, when it opportunistically flows between the humans and the technology. These two dimensions may serve to lay down the following performance levels (see Figure 3.2):

- **Role-based performance** – Encompassing routine tasks performed with mechanical control at the individual level. In this level any group activity is basically considered as a collection of independent activities.

- **Rule-based performance** – Concerning tasks accomplished with some latitude of decision from humans but within the constraints of a specific plan imposed by the technology. Unlike the previous level, the group activities are perceived as a collection of coordinated activities.

- **Knowledge-based performance** – Concerning interdependent tasks performed by humans in the scope of group and organizational goals.

This model highlights the increasing sophistication of human activity, where simple (from the organization’s perspective) individual roles are complemented with more complex coordinated activities and supplemented by even more complex knowledge-based and information-rich activities. The group becomes more important than the individual. We will use this model to delineate three distinct evaluation scenarios.

### 3.4.3. Evaluation Scenarios

There are three evaluation scenarios in the previously described three-layer view, which are the following:
3.4 Characterizing evaluation methods

Figura 3.2: Performance levels, adapted from [126]

**Role-based scenario**

The evaluation data is gathered at the individuals’ cognitive level, focusing on events occurring during a time frame in the order of minutes or even seconds. The most adequate evaluation methods to employ in this scenario adopt laboratory settings and considerable instrumentation (e.g., key logging). To gather the data, the evaluators must accurately specify the roles and activities; and the subjects must exactly act according to the instructions and under strict mechanical control. In these circumstances it is expected that the system detail is high (e.g., keystrokes and mouse movements) but the system scope is low (e.g., roles associated to some particular functions). This scenario also trades off realism towards higher precision and generalizability. The time invested in this type of evaluation tends to be low and mostly used in the preparation of the experiment. The various trade-offs associated to this evaluation scenario are illustrated in Figure 3.3.
3.4 Characterizing evaluation methods

Rule-based scenario

The evaluation data now concerns several subjects who must coordinate themselves to accomplish a set of tasks. The relevant events now occur between the several minutes and hours, instead of minutes or less. The system details being considered have large granularity (e.g., exchanged messages instead of keystrokes). The system scope also increases to include more functions. The evaluation methods employed in this scenario may still adopt laboratory settings although using less instrumentation. This scenario may still adopt laboratory settings although using less instrumentation. This scenario also represents trading off realism in favor of precision and generalizability. As with the role-based scenario, the evaluators must plan the subjects’ activities in advance; however, the subjects should be given more autonomy since control concerns the coordination level and not individual actions. The time invested in this type of evaluation is higher than the previous case, since the data gathering takes more time and the data analysis is less straightforward (e.g., requiring debriefing by the participants). The trade-offs associated to this evaluation scenario are illustrated in Figure 3.4.
3.4 Characterizing evaluation methods

Knowledge-based scenario

The evaluation data is mostly focused on the organizational impact and thus concerns much longer time frames, usually in the orders of days, months and even years, since the technology assimilation and the perception of value to the organization may take a long time to emerge and stabilize. The evaluation scenario is also considerably different when compared to the other scenarios, involving for instance knowledge management, creativity and decision-making abilities. Considering these main goals, it is understandable the system detail has coarse granularity, favoring broad issues such as perceived utility or value to business. The system scope may be wider for exactly the same reason. In this case the evaluators may not specify the roles and activities beforehand, as the subjects have significant latitude for decision, which leads to open situations beyond the control of the evaluators. Considering the focus on knowledge, the trade-off is usually to reduce the precision and generalizability in favor of realism. All these differences imply the laboratory setting is not the most adequate to the knowledge-based scenario, in favor of more qualitative settings. Two examples of evaluation methods employed in this scenario are case studies and ethnographic studies. These techniques need significant time to gather the data in the field for long periods, and also time to
3.4 Characterizing evaluation methods

transcribe, code and analyze the obtained data. The trade-offs associated to this evaluation scenario are illustrated in Figure 3.5.

![Figure 3.5: Knowledge-based scenario](image)

3.4.4. Classification of methods

Figure 3.6 shows the evaluation methods that were presented in Section 3.2.2 organized considering the role, rule and knowledge-based categories.

1. The knowledge-based evaluation emphasizes variables pertaining more to the organization and group than to the individual performance. Examples of metrics that can be delivered by these methods include interaction, participation, satisfaction, consensus, usefulness, and cost reductions.

2. On the contrary, the role-based evaluation stresses the importance of the individual performance. Metrics that can be obtained using these methods are efficiency and usability.

3. The rule-based evaluation may be seen as being in the middle of the extremes. Some metrics may include the organizational goals, e.g. conformance to
regulations, while others may concern group performance, such as productivity.

![Figure 3.6: Classification of evaluation methods](image)

Some methods may belong to one or two categories depending on the nature of the instruments involved in each method, e.g. cooperation scenarios (COS) are located in the area between rule and knowledge-based evaluation methods, because it has elements belonging to both categories. This classification allows evaluators to choose an appropriate method for their particular evaluation scenario.

### 3.5. Evaluation Cost

The evaluation cost is important during the selection of an evaluation method, since in some situations (e.g., when the system has to be developed in a short period of time), applying expensive methods may be unfeasible. However, a way to systematically compare evaluation costs had not been proposed, although some
methods are usually described as “discount”, or inexpensive. This is concerned with the second hypothesis of this thesis, \( H2: \) The cost of the proposed evaluation method will be similar to existing evaluation methods, according to the process duration and effort required to conduct it. This section explains the criteria for cost comparison.

After reviewing the existing methods, the evaluation cost was proposed to be a function of the process duration and the effort required to conduct the evaluation. Figure 3.7 displays a graph containing the previously presented evaluation methods, categorized according to their cost.

The effort to do an evaluation was estimated based on the activities that must be done and the required human resources. The activities that require the most effort are represented as small icons. An evaluation may require a prototype, which means developers or evaluators have to generate a reduced version of the final system to be able to conduct evaluation. \textit{Modeling} the collaborative system or process may also be needed, which generally requires a deep analysis of the situation. If an evaluation method requires a high number of users, they may have to be interrupted from their work to be interviewed or observed, which is costly. Some methods also require experts, who are specialized in a particular topic, and this also increments the complexity of the evaluation process. Naturally, these activities may be present in different intensities depending on the scenario - e.g., a method may require users for several focus groups and interviews, while another may only require users to fill out a short questionnaire. Other activities also impact the effort required to conduct evaluation. For these reasons, instead of assigning exact scores, the effort is classified in one of three categories: high, medium or low.

The duration of an evaluation method may be as short as a few hours, in the case of GHE, or as long as weeks, for QDE and GOT. In Figure 3.7, methods closest to the origin are those of lowest cost, while those in the upper right corner have the highest cost. The combination of time span and effort into a single cost measure depends on the particular situation.
3.6 Discussion

Evaluating collaborative systems is necessary and yet, many of them are not evaluated. Unevaluated systems tend to be unsuccessful because they may fail to consider the context, stakeholders and contain errors after deployment.

We proposed several classifications for evaluation methods that afford visibility to each one, allowing for fast comparison according to several criteria. We also proposed a way to characterize an evaluation method according to several relevant characteristics. These classifications provide a tool for any interested stakeholder to choose an evaluation method that is especially useful for his particular situation. The process of choosing an appropriate evaluation method is simplified, because a list of applicable methods is provided according to the needs of the stakeholder, product state, focus of the evaluation, etc. The evaluator may then choose the most appropriate method by reviewing his/her available resources (equipment, time, effort, etc) and the characteristics of each method.

There are also some limitations associated to categorizing evaluation methods according to a finite number of criteria. The criteria are static and cannot represent all of the possibilities of real world evaluation. For example, it is possible that the interested stakeholder is not a developer, user or the organization, or even, it is possible that the stakeholder has several roles (e.g., a user who is a manager at the organization). It could also be possible that an unforeseen characteristic is the most important one for a particular situation. Also, this categorization does not
suggest how to combine several methods to get a multi-faceted and more complete evaluation process. For these reasons, the lists are only a guide, and not a definitive answer for all possible situations. Chapter 4 discusses how to organize an evaluation process with more detail.

The categorization of evaluation methods suggested some areas that lack appropriate evaluation methods, providing opportunities for further research, such as in the case of developers who need to modify a finished product. The comparison of the reviewed methods also highlighted the fact that most evaluation methods are qualitative, and only two of them are purely quantitative. The prevalence of qualitative methods may be symptomatic of the complexity of collaborative systems, as human judgment may be required to disentangle the multiple contingencies, and ultimately appreciate if an application is good or not. On the other hand, the role of quantitative methods in evaluation is also important, since they permit the objective comparison of several applications, and may be automated. This suggests that new quantitative evaluation methods are needed. Further research in this area should improve the availability of methods for all stakeholders.

This chapter introduced criteria that can help compare the cost of several evaluation methods. For this, cost was defined to be a function of the time and effort required to do the evaluation. Time has three levels: hours, days, or weeks (or more, since some evaluation methods can be applied for months or years). Effort is determined as a combination of the work that must be done and resources that are needed, so four aspects are highly relevant: the need for users or experts, and whether the evaluation requires modeling or a prototype. Of course, each one of these aspects may have a varying weight, e.g. one method can require an intensive modeling of interface details, while another requires a simple modeling of use cases. For this reason, the person doing the classification can determine effort to be high, medium, or low, instead of an exact number. The combination of effort and time gives us an approximation of the cost of doing the evaluation - e.g., an evaluation that requires five users for five hours is less costly than an evaluation requiring five users for a full month. Chapter 5 introduces the proposed evaluation method and uses these criteria to compare its cost, and chapter 7 completes its
proof with empirical data. Naturally, it should be acknowledged that this is not the only possible way to compare evaluation costs. Other mechanisms could potentially be used to compare costs and may yield slightly different results.
3.6 Discussion

Using only one type of evaluation may prevent evaluators from gaining access to the complete picture in some cases. This suggests that several evaluation methods may be applied to obtain a comprehensive understanding of the system and its environment. CSCW systems are multifaceted. Conducting a thorough evaluation may provide additional perspective on how they function and how to improve them.

This chapter reviewed existing methods for collaborative systems. Mobile collaborative systems combine collaborative systems with the challenges of mobility. Therefore, although the methods presented here could be applied to mobile systems, they do not consider important aspects of mobility such as group composition variability, task flexibility, connectivity problems and low interdependence. This thesis proposes an evaluation method for these type of systems.
Capítulo 4
Evaluation Framework

The previous chapters reviewed existing evaluation methods, and presented strategies for their classification and selection. This chapter introduces a theoretical framework to organize the entire evaluation process of a collaborative system [59]. Then, the following chapter presents the evaluation method designed for mobile shared workspaces. This evaluation method can be used on its own, or inserted into an evaluation process that follows the guidelines presented here. The evaluation framework presented in this chapter is based on guidelines for the selection of evaluation methods according to the status of the product to be evaluated and the evaluation lifecycles that can be used to organize the evaluation process.

4.1. Evaluation Guidelines

This section presents a set of guidelines about the techniques and instruments used in collaborative systems evaluation. Figure 4.1 presents a summary of the recommended evaluation method categories considering the development status of the product to be assessed. These guidelines will help evaluators to select an appropriate evaluation method for each particular situation. The guidelines are briefly explained below.

1. If the product to be evaluated is in the conception stage (including analysis and design), then the evaluation should be oriented towards obtaining
coarse-grain information to help understand the role of the tool within the organization, the users’ expectations and needs, the business case and the work context. This information, usually obtained from knowledge-based evaluation methods, may be very useful to specify or refine the users and software requirements, to establish the system scope, to identify product/business risks and to validate a product design. To perform this assessment, evaluators should adopt an inter-subjectivist view over the collected data, considering qualitative and interactive ways to obtain data, using various activities such as field studies, focus groups, and meetings.

![Diagram of evaluation methods](image)

**Figura 4.1:** Summary of guidelines for selection of evaluation methods

2. If the product is in the implementation stage, a knowledge-based evaluation method is recommended. It will serve to understand if the product is able to deal with organizational goals. This evaluation also provides coarse-grain information concerning the issues/components requiring improvements. If the available time and budget allow additional evaluation actions, the process
may be complemented with a rule-based evaluation method. It will provide information necessary to adjust the product to the actual working scenario. For example, adjustments to concrete business processes may be identified this way. Optionally, a role-based evaluation method could also be used to fine-tune the product to the users. In case of rule/role-based assessments, the evaluation setting may be configured to assess the users’ activities in a controlled or mixed environment, which may utilize the laboratory. The evaluators may also adopt a more experimentalist view over the collected data.

3. An evaluation may also serve to determine the current impact a system into production has on its business operations. Therefore, the first recommended evaluation action considers diagnosing the current situation using precise information obtained from the actual production system. A role-based evaluation method may then be used. Similarly to the previous case, if the available time and budget allow additional evaluation actions, rule-based and knowledge-based methods could subsequently be applied. The aim is identifying concrete performance issues and improving organizational behavior.

4. Many organizations often decide to reengineer a legacy system. The main purpose is changing the organizational behavior by extending the system support. The existing system may be used to guide this reengineering. We recommend starting with a rule-based evaluation to avoid anchoring the evaluation on too fine-grained or too coarse-grained information. Nevertheless, a following knowledge-based evaluation may be done to determine the impact of the reengineered product on the organizational strategy. If the reengineering process involves significant changes to the systems’ functions, user interfaces or interaction paradigms, a role-based evaluation may also be recommended.

5. Often an evaluation action occurs when procuring a product. In such cases, we recommend starting the evaluation with a knowledge-based method, to understand if the system functionality matches the organizational needs.
Eventually, if the evaluators must also assess the system support to the organizational context and specific business processes, then the recommendation is to perform a rule-based evaluation.

Besides the generic recommendations mentioned above, the evaluators should also ponder the specific characteristics of the product under evaluation, namely the control and situation dimensions, which have impact on the evaluation scenario. The knowledge-based evaluation is naturally most adequate to products giving latitude of decision to the users and supporting interaction, collaboration and decision-making.

The evaluators should also ponder risk analysis. The risk adverse evaluator will set up a complete evaluation process, considering a combination of the three evaluation types, starting with the knowledge-based and finishing with the role-based scenarios. The risk taker evaluator will probably concentrate the evaluation only on the knowledge-based issues. The payoff of this high-risk approach is streamlining the evaluation efforts while focusing upon the issues that may have highest impact on the organization. The associated risk is the potential lack of quality of the outcomes.

4.2. Evaluation Lifecycle

The analysis of the possible scenarios highlights interesting issues to ponder when undertaking a collaborative system evaluation process. Regarding the ensemble of variables, the rule-based scenario seems to be the most balanced in the adopted compromises. On the contrary, the role-based and knowledge-based scenarios show a clear tendency for the extremes. The role-based scenario emphasizes detail, precision, generalization and time at the prejudice of scope and realism. On the opposite side, the knowledge-based scenario shows a clear emphasis in scope and realism at the expense of detail, precision, generalization and time. These differences highlight the so-called instrumentalist and inter-subjectivist strategies,
which have been quite influential in the CSCW field [112, 99, 57]. The instrumentalist strategy is mostly focused on accumulating knowledge through experimentation, while the inter-subjectivist strategy is rather concerned with interpreting the influences of the technology on the individuals, groups and the organization.

The analysis of some individual variables may also give additional insights about the collaborative system evaluation. One such variable is invested time, which is distinct for the three discussed scenarios. From a very pragmatic perspective, the selection of the evaluation scenarios could be based on the time one is willing to invest on the evaluation process. Such considerations would lead to a preference for the rule-based and role-based scenarios and a devaluation of the knowledge-based scenarios. Nevertheless, this approach may not be feasible due to lack of system detail, e.g. whenever evaluating design ideas. This approach also has some negative implications, such as emphasizing details of little importance to the organization.

We should also consider that system detail and scope are related with the strategy selected to bring design ideas towards an actual running system. For instance, a breadth-first strategy indicates a strong initial focus on broad functionality, which would mandate an evaluation starting with a knowledge-based scenario and later on continuing with role-based and rule-based scenarios. On the contrary, a depth-first strategy indicates a strong preference for fully developing a small functional set, which would mandate an evaluation starting with a role-based scenario and proceeding with rule-based and knowledge-based scenarios.

Figure 4.2 provides an overview of these evaluation issues. The two dotted lines show the limits suggested by the three evaluation scenarios. The arrows show the possible directions of the evaluation strategy and their basic assumptions. The arrows in Fig. 4.2 indicate possible evaluation processes adopted according to various biases. The arrow’s start point indicates which type of evaluation should be done first, while the end point suggests where to finish the evaluation process.
This graphical representation also affords equating the collaborative system evaluation in other dimensions. For instance, the specific control and situation characteristics of one particular system may determine the effort involved in evaluation. Consider e.g. a system under evaluation which only supports mechanical control. Then, we may consider the low dotted line shown on Figure 4.2 corresponds to the most adequate evaluation, and thus it should be centered on role-based performance. If the case where the system being evaluated concerns routine and planned activities, then the dotted line should be moved to the upper-left side and the evaluation should then be focused on the role and rule-based performance.

4.3. Discussion

This chapter provides ideas about how to organize an entire evaluation process, suggesting how to select evaluation methods that will provide evaluators information at several levels of detail. Naturally, the evaluation process has to
deal with conflicting forces, such as the cost of the evaluation process and the level of understanding that can be gained from evaluation. On one hand, the cost of the evaluation process, which is related to the invested time and effort, can lead evaluators to select the lowest-cost methods, or to use only one evaluation method, which can give only a partial picture of the software. On the other hand, the level of understanding to be gained from evaluation increases as more methods at varying levels of detail are used. This will naturally increase the costs of the evaluation process. As has been mentioned, precision, generalizability and realism are difficult, if not impossible, to maximize simultaneously. To do so would provide greater understanding while increasing the cost of evaluation. Evaluators have to balance these conflicting forces to carry out an evaluation process that is adequate to the setting, i.e., they have to use the most appropriate evaluation methods for the time and effort they are willing to invest in the evaluation process.

These problems perhaps cannot be solved completely. However, providing developers with more low-cost evaluation methods can allow them to conduct an evaluation process that provides the required outcomes without requiring too many resources. The methods presented in the previous chapter generally focus on collaborative systems, which means they could also be applied to mobile collaborative systems, even though they lack relevant information about the particular aspects of mobility. The framework presented here can also be applied to mobile collaborative evaluation, since this would only require a characterization of the evaluation method in terms of the aspects detailed here.

The next chapter introduces the evaluation method proposed in this thesis for mobile collaborative systems. This method can be used to provide an assessment of the collaborative capabilities of a mobile collaborative application, as well as suggestions for improvement. Naturally, it can be incorporated into the framework presented here, providing a start on methods specifically tailored to mobile collaborative systems evaluation.
Capítulo 5

Mobile Collaboration Evaluation (MCE) Method

This chapter introduces the MCE (Mobile Collaboration Evaluation) method. This method is based on the requirements presented in Chapter 2 and on a modeling language called the MCM modeling language that will be presented in detail in Chapter 6. This chapter first describes the steps to carry out an evaluation using this method. Then, the evaluation method is discussed with respect to the characterization presented in the previous chapter. Finally, we present the cost of the evaluation method and a brief discussion.

5.1. MCE Method

The MCE method is a formative evaluation method aimed towards evaluating the support for collaborative processes a mobile application provides. This is accomplished through a list of functional requirements to support collaboration, which is automatically generated by providing a description of the mobile collaborative work through a modeling language. This modeling language - the MCM language - provides graphic elements to describe the collaborative process. The resulting graph can be combined with the previously presented list of requirements to obtain the suggested functional requirements checklist. The results of the evaluation method should be fed back into the development process to improve the
5.2 Evaluation Method Characterization

The steps towards conducting evaluation using MCE are the following ones. First, evaluators must use the MCM language to build an MCM graph by following the steps described in the next chapter, which involve observing and interviewing workers. The obtained knowledge must be used to understand the collaborative work that can be done through the system that is being designed. The MCM graph will describe the actors and the collaborative process they will engage in during their work through the system. Then, the evaluators must determine if any general requirements do not apply to the collaborative work scenario. They may remove any requirement that does not apply to the particular scenario they will be evaluating. Then, the MCM graph describing a collaboration process is combined with the revised list of requirements to generate a list of suggested requirements for the particular situation. The requirements are produced according to the scenarios in which the two involved actors interact. Finally, an application to support the evaluation process will automatically generate items to check or test. Evaluators must review the application, marking the successful tests, the unsuccessful ones and providing any additional comments that may be helpful to developers when modifying the application. A score (the percentage of tests that were successful) may be calculated, and saved to be compared with later applications of the evaluation method. Naturally, evaluation results should be communicated within the development team and used to redesign and improve the application. Table 5.1 summarizes the steps of the MCE method.

5.2. Evaluation Method Characterization

As discussed in chapter 3, there are tradeoffs involved in evaluation, and no single evaluation method can simultaneously maximize precision, realism and generalizability. In case of the proposed method, we are deliberately opting to decrease realism in favor of precision and generalizability. This means the evaluation method is easy to implement (it is not necessary to conduct evaluation in the actual work setting or to interrupt users from their work). MCE can be categorized as a
Cuadro 5.1: Steps for MCE

<table>
<thead>
<tr>
<th>Step</th>
<th>Goal</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.1</td>
<td>Create MCM Graph</td>
<td>Observation, interviews, modeling</td>
</tr>
<tr>
<td>E.2</td>
<td>Determine general requirements</td>
<td>Specify which general requirements do not apply</td>
</tr>
<tr>
<td>E.3</td>
<td>Generate tests list</td>
<td>Software automatically generates tests</td>
</tr>
<tr>
<td>E.4</td>
<td>Software revision</td>
<td>Evaluate and mark successful and unsuccessful tests and any additional comments</td>
</tr>
</tbody>
</table>

rule-based evaluation method, which has a tradeoff of realism in favor of precision and generalizability. The rest of the characteristic variables can be described as follows. MCE is concerned with detail at a high level of granularity, since it deals with functional requirements (e.g. messaging and awareness). The time invested is medium, since observation and a modeling phase are required. Finally, the system scope is also medium, since the functions being assessed are several, but not exhaustive. These features make the MCE method have the shape shown in Figure 5.1 in terms of our framework.

This characterization allows us to insert the MCE method into the framework described in chapter 4. Rule-based evaluation methods are in the middle ground between role-based methods, which have a high level of detail and focus on the individual, and knowledge-based methods, which focus on organizational issues and have coarse granularity. This results in rule-based methods being applicable over a wider range of situations, which may promote their use. According to the presented guidelines, it is most appropriate to use this method to begin an evaluation process in case of a product being developed that is under reengineering, or a product being acquired that is known. Naturally, it is also appropriate to use this method in several other scenarios, e.g. while simultaneously applying a long-term knowledge-based evaluation method.
5.3. Evaluation Method Cost

The evaluation cost is important during the selection of an evaluation method. Chapter 3 introduced a way to estimate the cost of an evaluation method based on the duration of the process and the effort required to conduct it. This section theoretically estimates the cost of MCE to begin the proof of H2, which is concerned with the cost of MCE in relation to existing evaluation methods. In case of the MCE method, the time span will be in the range of days, because observation and analysis of the work scenario are required before the modeling can be done. The effort is related to the required modeling, but users and experts are not necessarily involved. Therefore, the cost of MCE is relatively low and comparable to inexpensive evaluation methods such as GWA and CUA. Figure 5.2 depicts the cost of all the evaluation methods we have mentioned, including MCE.

Although the cost of MCE appears to be low, this should be confirmed with empirical data. Chapter 7 further discusses the validity of H2 by presenting expe-
5.4 Discussion

The MCE method was designed to deal with several issues that make evaluation difficult. First, this method is formative and can therefore be applied at any stage of the development cycle of a product. This means that a design, a prototype or a fully functional application may be evaluated. The method does not require users nor experts; it only needs analysts and developers who are part of the development team. In this way, the cost of applying the evaluation method is kept reasonably low. This does not mean to undermine the importance of users in the development of an application: they may still be central to development, but the MCE method can be used to generate ideas to contrast and compare with users’ opinion, especially since users typically are not aware of the collaborative support an application should offer.

Naturally, the MCE method has some limitations. First, this method focuses on the compliance of an application to a list of functional requirements. Therefore, it does not evaluate other important aspects of software, such as usability and performance. Second, it also focuses on the group level, and does not address individual or organizational requirements. It is not costless either, since it does require
an understanding of the environment, which can be acquired through interviews and observation, and it also requires modeling effort. However, we have tried to keep the costs as low as possible by not requiring users, a prototype, or experts, and also by computerizing the evaluation process as much as possible.

The next chapter details the main part of the MCE method, which is the MCM modeling language. The combination of an MCM graph and the list of requirements presented in chapter 2 are the central parts of this evaluation method.
Capítulo 6

Mobile Collaboration Modeling (MCM) Language

This chapter presents the modeling language developed in this thesis, which is targeted specifically at mobile collaborative systems. The first section presents a brief summary of the related work. Then, the next section formally defines the language, including the validation mechanisms to ensure it presents a coherent collaboration scenario. Section 6.3 briefly describes the software tool that was developed to support modeling, and finally, some of the benefits and limitations of this proposal are discussed.

6.1. Related Work

There exist several techniques to model software (including collaborative systems and loosely coupled systems) intended to support system design. An initial proposal is the unified modeling language (UML); it is a general purpose modeling language that is used for software engineering. UML use case diagrams allow designers to identify actors and the roles they play in the system; activity diagrams describe control flow. There are several other modeling techniques, such as data flow diagrams and agile modeling. However, all of these techniques are designed for general use and lack ways to represent collaborative and mobile aspects of work.
6.1 Related Work

Constructing models for collaboration using these tools would require adapting them to collaborative work. This adaptation, though possible, is a complex process that would force a language to be used in ways its designers had not foreseen.

Several modeling techniques for collaborative systems focus on task analysis. Two well-known techniques are Groupware Task Analysis (GTA) [145] and Collaboration Usability Analysis (CUA) [118]. GTA aims to model the task domain for collaborative systems, specifying agents, tasks, objects and the environment. CUA is oriented towards collaboration, representing individual tasks and periods of teamwork, and displaying loosely coupled work. However, CUA is low-level and very detailed; thus, it is useful to design specific details of a system, but it requires much modeling effort and it does not provide an overview of the collaborative process. Other techniques for collaborative system modeling are those based on Petri nets and interactive cooperative objects [109], and a colored Petri net extension of task-based approaches using UML [45]. There are also formal models such as input/output automata for dynamic systems [8] and agent-based models [78].

However, these modeling techniques alone may not be enough to model loosely coupled mobile group work. This type of work has additional complexity because of its flexibility in location, connectivity and degree of involvement in the collaboration. Pinelle [113] reviews several ways to model mobile loosely coupled systems, such as sociograms, contextual design work models and CUA. Pinelle [113] proposes an analysis technique for loose coupling based on them to provide an overview of important features of the collaboration scenario. This technique was developed while working on a mobile collaborative application for home care workers. It consists of an interaction model, awareness model, coordination model, group task model and loose coupling checklist. Each model covers a different aspect of work and captures relevant information for system design. This is the closest to the approach presented here; however, we focus on giving an overview of the whole process and on some key issues of mobility such as connectivity.
6.2. Modeling language definition

Understanding and describing mobile collaborative work is difficult, due to the mobility of the involved actors in time and space. A graphical modeling language was defined to help solve this problem. This section introduces this language, called Mobile Collaboration Modeling (MCM) Language, as a formal graphical nomenclature that may be used to model loosely coupled mobile collaboration [61].

6.2.1. Language Specification

The Mobile Collaboration Modeling language is based on a directed graph $G=(N,A)$. The nodes represent the roles of the actors participating in the collaboration process; they can either be active users or intermediaries in the collaboration scenario (Figure 6.1). Intermediaries are passive participants that enable others to collaborate, e.g., a server allowing users to synchronize their work. Arcs are used to represent interaction between two roles. A directed arc from A to B represents that role A may send information to role B (Figure 6.2); otherwise, if no collaboration is considered, then there will be no line between them. Although the direction of each arc captures important information about the collaborative process, undirected graphs are used to analyze MCM graphs. The reason for this is shown in Lemma 6.2.3.

![Figure 6.1: Types of nodes](image-url)
6.2 Modeling language definition

Figura 6.2: Types of arcs

Cuadro 6.1: Label representations - simple labels

<table>
<thead>
<tr>
<th>Label</th>
<th>Meaning</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR</td>
<td>Simultaneous Reachable</td>
<td>![SR Representation]</td>
</tr>
<tr>
<td>SU</td>
<td>Simultaneous Unreachable</td>
<td>![SU Representation]</td>
</tr>
<tr>
<td>NR</td>
<td>Non-simultaneous Reachable</td>
<td>![NR Representation]</td>
</tr>
<tr>
<td>NU</td>
<td>Non-simultaneous Unreachable</td>
<td>![NU Representation]</td>
</tr>
</tbody>
</table>

Each arc is labeled to represent the possible scenarios in which interaction will take place, which correspond to the classification scheme described in section 2.1. Each arc may have a simple label (SR for simultaneous-reachable, SU for simultaneous-unreachable, NR for non simultaneous-reachable or NU for non simultaneous-unreachable), represented in Table 6.1, or a composite label, which is an OR composition of simple labels and is represented by a square with the corresponding colored quadrants. For example, a representation showing a colored top half means a SR OR SU composite label. This means that when one of the actors requires to interact with the other, their relationship will be simultaneous and either reachable or unreachable.

Lemma 6.2.1. There are 15 possible labels on an arc.

Demostración. The possible simple labels are 4. A composite label is a combination of one or more simple labels, so we may calculate the number of labels by
6.2 Modeling language definition

calculating the possible combinations of chosen labels. For this, we use the binomial coefficient formula for choosing \( k \) elements over \( n \), which is the following:

\[
\binom{n}{k} = \frac{n!}{(n-k)!k!}
\]

- For \( k=1 \), there are \((4!)/(3!1!) = 4\) combinations.
- For \( k=2 \), there are \((4!)/(2!2!) = 6\) combinations.
- For \( k=3 \), there are \((4!)/(1!3!) = 4\) combinations.
- For \( k=4 \), there is 1 combination. Therefore, in total there are \( 4+6+4+1 = 15 \) combinations.

6.2.2. MCM graph analysis

This section presents the characteristics of MCM graphs and it proposes analysis mechanisms to help determine the validity of an MCM graph. The analysis is organized in three phases: first, properties of graphs consisting of two nodes are presented (section 6.2.2.1). Next, graphs containing cycles are discussed, divided in two cases: cycles of exactly 3 nodes, or triangles (section 6.2.2.2), and cycles of more than 3 nodes (section 6.2.2.3). In this analysis, we consider relationships between two nodes (1-1 relationships) and not relationships between a node and any number of nodes (1-m relationships) since we may instead choose to view this situation as \( m \) 1-1 relationships (Figure 6.3).

![Diagram of MCM graphs](image)

Figure 6.3: Equivalence between (a) 1-m and (b) 1-1 relationship analysis
6.2 Modeling language definition

6.2.2.1. Two-node analysis

When two roles $r_1$ and $r_2$ interact, they do so in a given collaboration scenario and a given direction of information flow (Figure 6.4).

![Figura 6.4: Interaction between two nodes](image)

**Definition 6.2.2.** The scenarios of interaction two roles pass through while one of them is active are called the possible interaction scenarios between roles $r_i$ and $r_j$. We note them $s_{i,j}$.

**Lemma 6.2.3.** The direction of the interaction does not affect the interaction scenarios between two roles.

**Demostración.** The direction of interaction represents the flow of information from one role to another. In some cases, both users will require sending information to each other, while in others (e.g., a user submitting information to a server) one role will act only as a sender. The scenarios in which both users are at a given moment do not depend on whether they intend to send, receive or send and receive information.

Direction of information flow may be important when designing mobile applications, but as a consequence of Lemma 6.2.3, we change the MCM graphs to undirected graphs for their analysis from now on.

6.2.2.2. Three-node cycle analysis

When three roles collaborate among themselves, we study the scenarios of interaction of the three relationships to ensure consistency. In Fig. 6.5, roles $r_1$, $r_2$ and $r_3$ interact, forming cycle $r_1r_2r_3r_1$. If we know how $r_1$ and $r_2$ interact, and how $r_2$ and $r_3$ interact, we may infer some information about the interaction between...
6.2 Modeling language definition

$r_1$ and $r_3$.

**Definition 6.2.4.** A cycle of length 3 $r_1 r_2 r_3 r_1$ is called a triangle, and it is noted $r_{123}$.

![Diagram of a triangle with vertices $r_1$, $r_2$, and $r_3$, and edges $s_{1,2}$, $s_{1,3}$, and $s_{2,3}$]

**Figura 6.5:** Triangle $r_{123}$

**Lemma 6.2.5.** Considering triangle $r_{123}$, if $s_{1,2}$ is simultaneous, and $s_{2,3}$ is simultaneous, then $s_{1,3}$ is also simultaneous.

**Demostración.** If $r_1$ and $r_2$ work simultaneously, this means that during all the time in which $r_1$ is active, $r_2$ is also active. If $r_2$ and $r_3$ are simultaneous, this means that during all the time in which $r_2$ is active, $r_3$ is also active. That is, while $r_2$ is active, both $r_1$ and $r_3$ must also be active, which means $r_1$ and $r_3$ are also simultaneous.

**Lemma 6.2.6.** Considering triangle $r_{123}$, if $s_{1,2}$ is simultaneous, and $s_{2,3}$ is non-simultaneous, then $s_{1,3}$ is non-simultaneous.

**Demostración.** Since $r_1$ and $r_2$ are simultaneous, if $r_1$ and $r_3$ were simultaneous, through Lemma 6.2.5 we would conclude that $r_2$ and $r_3$ must also be simultaneous. This is a contradiction because $r_2$ and $r_3$ are non-simultaneous. Therefore, $r_1$ and $r_3$ must also be non-simultaneous.

**Lemma 6.2.7.** Considering triangle $r_{123}$, if $s_{1,2}$ is non-simultaneous, and $s_{2,3}$ is non-simultaneous, then $s_{1,3}$ may either be simultaneous or non-simultaneous.
Demostración. If $r_1$ and $r_2$ are non-simultaneous, this means that both are active at different times. If $r_2$ and $r_3$ are non-simultaneous, this means that both are active at different times. In this case, there are two possibilities:

- $s_{1,3}$ may be simultaneous, which means $r_1$ and $r_3$ work at the same time, but at a different time from $r_2$.
- $s_{1,3}$ may be non-simultaneous, which means $r_1$, $r_2$ and $r_3$ work all work at different time periods.

Lemma 6.2.8. Considering triangle $r_{123}$, if $s_{1,2}$ is reachable, and $s_{2,3}$ is reachable, then $s_{1,3}$ is also a reachable connection.

Demostración. If $s_{1,2}$ is reachable, this means that $r_1$ may communicate with $r_2$ in a highly predictable way. If $s_{2,3}$ are reachable, $r_2$ may communicate with $r_3$ in a predictable way. Therefore, $r_1$ and $r_3$ are available and there is a communication channel that can connect them, possibly through $r_2$.

Lemma 6.2.9. Considering triangle $r_{123}$, if $s_{1,2}$ is a reachable connection, and $s_{2,3}$ is an unreachable connection, then $s_{1,3}$ is unreachable.

Demostración. If $s_{1,2}$ is reachable, this means $r_1$ may communicate in a predictable way with $r_2$. If $s_{1,3}$ were reachable, because of Lemma 6.2.8, then $s_{2,3}$ would be reachable. Since $s_{2,3}$ is an unreachable connection, $s_{1,3}$ must be unreachable as well.

Lemma 6.2.10. Considering triangle $r_{123}$, if $s_{1,2}$ is an unreachable connection, and $s_{2,3}$ is an unreachable connection, then $s_{1,3}$ may be either reachable or unreachable.

Demostración. If $s_{1,2}$ are unreachable, and $r_2$ and $r_3$ are unreachable, then there are two possibilities:

- $s_{1,3}$ may be reachable, in case e.g. in case $r_2$ is unavailable but $r_1$ and $r_3$ are not
- $s_{1,3}$ may be unreachable, if all three roles are disconnected from each other, e.g. due to lack of a communication channel.
6.2 Modeling language definition

Cuadro 6.2: Consistency Rules

<table>
<thead>
<tr>
<th>N</th>
<th>s₁₂</th>
<th>s₂₃</th>
<th>⇒</th>
<th>s₁₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>⇒</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>⇒</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>⇒</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>⇒</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>⇒</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>⇒</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>⇒</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>⇒</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td>⇒</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>⇒</td>
<td></td>
</tr>
</tbody>
</table>

**Lemma 6.2.11.** Considering a triangle \( r_{123} \), we may infer the possible scenarios of interaction \( s_{1,3} \) if \( s_{1,2} \) and \( s_{2,3} \) are known and each consists of only one possible scenario. This is shown in Table 6.2.

**Demostración.** Each scenario \( s_{1,2} \) and \( s_{2,3} \) consists of two dimensions: reachability and simultaneity. Given this and the fact that \( s_{1,2} \) and \( s_{2,3} \) must consist of only one interaction scenario, \( s_{1,2} \) and \( s_{2,3} \) each have four possible configurations. Therefore, we have ten distinct combinations of \( s_{1,2} \) and \( s_{2,3} \). To find \( s_{1,3} \) we use Lemmas 6.2.5, 6.2.6 and 6.2.7 for the simultaneity dimension, and Lemmas 6.2.8, 6.2.9, and 6.2.10 for the reachability dimension.

**Theorem 6.2.12.** **Triangle Inference.** Considering a triangle \( r_{123} \), we may
infer the possible scenarios of interaction between $r_1$ and $r_3$ if relationships $s_{1,2}$ and $s_{2,3}$ are known.
6.2 Modeling language definition

Cuadro 6.3: Example

<table>
<thead>
<tr>
<th>N</th>
<th>$s_{1,2}$</th>
<th>$s_{2,3}$</th>
<th>$\Rightarrow$</th>
<th>$s_{1,3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>$\Rightarrow$</td>
<td>$?$</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>$\Rightarrow$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>$\Rightarrow$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>$\Rightarrow$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>$\Rightarrow$</td>
<td></td>
</tr>
</tbody>
</table>

Demostración. There are two cases:

- $s_{1,2}$ and $s_{2,3}$ consist each of only one possible interaction scenario. This case was proved in Lemma 6.2.11.

- $s_{1,2}$ and/or $s_{2,3}$ consist of several possible interaction scenarios.

In this case, if $s_{1,2}$ or $s_{2,3}$ consist of several interaction scenarios, at a given point in time $r_1$ and $r_2$ may be at any scenario of the set $s_{1,2}$ and $r_2$ and $r_3$ may be at any scenario of the set $s_{2,3}$. We may find all possible situations by combining all elements (single interaction scenarios) of $s_{1,2}$ and all elements of $s_{2,3}$. This results in a maximum of 10 combinations of two scenarios (the case where $s_{1,2}$ and $s_{2,3}$ each include all 4 interaction scenarios). The resulting possible interaction scenarios for each combination may be obtained through Lemma 6.2.11. The union of all possible $s_{1,3}$ will be the resulting possible interaction scenarios for $r_1$ and $r_3$. □

For example, in triangle $r_{123}$, if $s_{1,2} = \{SR, NR\}$ and $s_{2,3} = \{SU, NU\}$, Table 6.3 presents the possible combinations of elements from $s_{1,2}$ and $s_{2,3}$. For each combination, Table 6.2 may be used to find the possible scenarios for $s_{1,3}$. The union of all possible resulting scenarios is $s_{1,3} = \{SU, NU\}$. 

92
6.2 Modeling language definition

6.2.2.3. n-node cycle analysis

When several roles are present in a collaboration scenario, we may study the scenarios of interaction in cycles of length greater than 3 to ensure consistency. In Figure 6.6 there is a path $r_1 r_2 r_3 \ldots r_{N-1} r_N r_1$. If we know the scenarios of interaction of the first N-1 edges, we may infer some information about the interaction scenarios of the final edge $r_N r_1$.

**Lemma 6.2.13.** Assume there is a path $r_1 r_2 r_3 \ldots r_{N-1} r_N$ in which $s_{i,i+1}$ for $i \in (1, \ldots, N-1)$ edge is simultaneous. If edge $r_N r_1$ exists, then it must also be simultaneous.

**Demostración.** If $r_1$ is simultaneous with $r_2$ and $r_2$ is simultaneous with $r_3$, then through Lemma 6.2.5 $r_1$ must be simultaneous with $r_3$. Then, $s_{1,3}$ is simultaneous. Since $s_{3,4}$ is simultaneous, through Lemma 6.2.5, $s_{1,4}$ is simultaneous. By induction on $i \in (2, \ldots, N)$, each $r_i$ will be simultaneous to $r_1$. So finally, $s_{1,N}$ (which is equivalent to $s_{N,1}$) will also be simultaneous. \qed

![Figure 6.6: n-node cycle](image)

**Lemma 6.2.14.** Assume there is a path $r_1 r_2 r_3 \ldots r_{N-1} r_N$ in which $s_{i,i+1}$ for $i \in (1, \ldots, N-1)$ edge is reachable. If edge $r_N r_1$ exists, then it must also be reachable.

93
6.2 Modeling language definition

Cuadro 6.4: Reglas para ciclo n-nodo

<table>
<thead>
<tr>
<th>N</th>
<th>( s_{i,i+1}, i \in (1,N-1) )</th>
<th>( \Rightarrow )</th>
<th>( s_{N,1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>or</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>or</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Demostración. Si \( s_{1,2} \) es reachable y \( s_{2,3} \) es reachable, entonces a través de Lemma 6.2.8 \( s_{1,3} \) debe ser reachable. Desde \( s_{3,4} \) es reachable, a través de Lemma 6.2.8 tenemos que \( s_{1,4} \) es reachable. Por inducción sobre \( i \in (2,...,N) \), cada \( r_i \) será reachable de \( r_1 \). Así que finalmente, \( s_{1,N} \) (que es equivalente a \( s_{N,1} \)) también será reachable.

**Theorem 6.2.15. Cycle Inference.** Considerando un ciclo \( r_1 r_2 r_3 \ldots r_{N-1} r_N r_1 \), podemos inferir los posibles escenarios de interacción \( s_{N,1} \) en algunos casos si \( s_{i,i+1} (i \in (1,...,N-1)) \) son conocidos. Los tres casos en los que \( s_{N,1} \) puede ser inferido se muestran en la Tabla 6.4, y son los siguientes:

- Si todos \( s_{i,i+1} \) son simultáneos y reachable, entonces \( s_{N,1} \) también será simultáneo y reachable.
- Si todos \( s_{i,i+1} \) son simultáneos y cualquiera sea reachable o unreachable, entonces \( s_{N,1} \) también será simultáneo y puede ser reachable o unreachable.
- Si todos \( s_{i,i+1} \) son reachable y cualquiera sea simultáneo o non-simultáneo, entonces \( s_{N,1} \) también será reachable y puede ser simultáneo o non-simultáneo.

Demostración. Cada escenario \( s_{i,i+1} \) consiste en dos dimensiones: reachability y simultaneity. Para el primer caso, dados los Lemmas 6.2.6 y 6.2.7 sabemos que si todos los escenarios \( s_{i,i+1} (i \in (1,...,N-1)) \) son simultáneos y reachable, escenario \( s_{1,N} \) también debe ser simultáneo y reachable. Para el segundo caso, sabemos que si todos los escenarios \( s_{i,i+1} (i \in (1,...,N-1)) \) son simultáneos, \( s_{1,N} \) también debe ser simultáneo, pero si algunos \( s_{i,i+1} \) son unreachable, \( s_{1,N} \) puede ser reachable o unreachable. Finalmente, para el tercer caso, sabemos que si todos los escenarios \( s_{i,i+1} (i \in (1,...,N-1)) \) son reachable, \( s_{1,N} \) debe ser reachable, pero si algunos \( s_{i,i+1} \) son non-simultáneos, \( s_{1,N} \) puede ser simultáneo o non-simultáneo.
6.2 Modeling language definition

6.2.3. MCM Validation

This section discusses how to apply the analysis presented in Section 6.2.2 to a complex graph representing a real-life situation, with several nodes and edges. The purpose of analyzing an MCM graph is to find inconsistencies and errors in it. We may identify which triangles and cycles are inconsistent, but it is not possible to distinguish exactly which interaction scenarios $s_{i,j}$ are erroneous. It will be up to the user to analyze the problematic cycle and to correct it.

6.2.3.1. MCM-V Algorithm

The MCM Validation (MCM-V) algorithm is used to conduct the validation. This algorithm is now presented as pseudocode.

BEGIN MCM-V ALGORITHM
Graph graph = getGraph();
Vector<Triangle> T = graph.getTriangles();
for each triangle t in T
    if (! t is valid according to rules)
        report inconsistency in triangle t
end if
end for
Vector<Cycle> C = graph.getCycles(); //excludes triangles
for each cycle c in C,
    for i = 1, ..., N
        if (! s(i,i+1) is valid given path r(i+1,...,i))
            report inconsistency in cycle c
        end if
    end for
end for
END
6.2 Modeling language definition

6.2.3.2. MCM-V Analysis

To analyze the MCM-V algorithm’s complexity, it is divided in two parts: first, we need to find all the cycles of the graph. Then, each cycle is analyzed to find errors or inconsistencies.

The complexity of the algorithm to find all simple cycles in an undirected graph is $O((N+E)C)$ [85], where $N$ is the number of nodes, $E$ the number of edges, and $C$ the number of cycles.

When all the cycles have been found, they are separated in two groups: cycles of length 3 and cycles of a longer length. For each triangle $t_{123}$, the following is checked.

- The combination of $t_{1,2}$ and $t_{2,3}$ can result in $t_{1,3}$. For this, all possible combinations of $t_{1,2}$ and $t_{2,3}$ are generated (at most 10 combinations), and for each table of 10 consistency rules is searched for the corresponding one. All the results are combined into one and then checked to see whether $t_{1,3}$ is contained in this generated result. Therefore, the complexity of this is a constant $K$.
- The combination of $t_{2,3}$ and $t_{1,3}$ can result in $t_{1,2}$. The algorithm is the same as in the previous point, so the complexity is again $K$.
- The combination of $t_{1,2}$ and $t_{1,3}$ can result in $t_{2,3}$. The algorithm is the same as in the previous point, so the complexity is again $K$.

In the case of cycles of length greater than 3, the consistency check looks at the relationships between all adjacent nodes, counting how many of each type exist. After this process, the rules presented in 6.2.15 are checked. The worst case is if the cycle visits all the nodes in the graph, so the complexity in the worst case is $N$.

Therefore, MCM-V has the following complexity:

- Finding all cycles: $O((N+E)C)$
- Checking consistency of each triangle: $O(K)$
6.2 Modeling language definition

- Checking consistency of each cycle of length greater than 3: O(N)

Finally, the total complexity is O(C(2N+E+K)). Naturally, each graph will have a number of nodes equal to the number of roles present in the collaborative process. This number can range from 1 to a much bigger number in case of a complex collaborative process, but from personal experience and other experiences in the literature, we foresee graphs generally having a few dozen nodes at most.

6.2.4. Modeling language use

The purpose of the MCM graph is to help developers understand a work scenario. The graph will present a clear overview of the collaboration scenario, and may therefore be used to communicate work practices, determine requirements and design an appropriate application. This section describes how to generate an MCM graph modeling a collaborative work scenario.

First, the roles involved in the collaborative work scenario must be identified, through interviews and observation of the work setting. From this set of roles, the ones that interact through the collaborative system will be present in the MCM graph. The identified roles must include any servers that are used to share and synchronize work, as they will be passive actors in the collaboration process.

Second, each role is characterized, detailing information which will be relevant to the collaborative work. The following information must be described: type(s) of mobile device that are used (e.g., PDAs, laptop computers, desktop computers), type(s) of network access available (e.g., ad-hoc networks, 3G, cable, none), working hours (if they can be established), and location(s) of work.

Then, for each role, developers must note its information flows (information that each role sends and receives, and the interacting nodes. This information is then used in the fourth step to compose the entire graph, consisting of all the interactions of the working group. Finally, the graph is validated using the MCM-V
6.3. Software Tool

To support the evaluation process, a software tool named *Graph Modeling Tool (GMT)* that allows developers to model a mobile collaboration setting through an MCM graph and to conduct evaluation. The GMT tool was implemented in Java 1.6, with the Prefuse Visualization Toolkit (http://prefuse.org/), using the NetBeans development environment.

The GMT tool has two main functionalities, which are:

- The creation of MCM graphs and their validation.
- Support for evaluation using the MCE method.

### Cuadro 6.5: Steps for MCM graph generation

<table>
<thead>
<tr>
<th>Step</th>
<th>Goal</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.1</td>
<td>Identification of roles</td>
<td>System users, servers</td>
</tr>
<tr>
<td>G.2</td>
<td>Role characterization</td>
<td>Location, Devices, Working</td>
</tr>
<tr>
<td>G.3</td>
<td>Identification of relationships</td>
<td>Information flows</td>
</tr>
<tr>
<td></td>
<td>between two roles</td>
<td></td>
</tr>
<tr>
<td>G.4</td>
<td>Graph composition</td>
<td>Identified relationships</td>
</tr>
<tr>
<td>G.5</td>
<td>Graph validation</td>
<td>Composed graph</td>
</tr>
</tbody>
</table>

With this characterization, developers may construct a graph identifying roles and directional communication and collaboration between them. For each edge between two roles in the graph, developers should study the types of mobile devices, locations of work, communications infrastructure and working hours to find whether collaboration is reachable or unreachable and simultaneous or non-simultaneous. A summary of the steps that should be followed to construct an MCM graph is presented in Table 6.5.
If a development team is in the beginning stages of developing a MSW, one of their first tasks will be to understand the setting and the collaborative work that is done and that should be supported by the mobile application. The MCM language can help the developers model and understand the setting. For this, a developer can create a graph through the GMT tool, by adding roles (Fig. 6.7a), relationships between them (Figure 6.7b) and editing the scenarios of interaction (Figure 6.7c). The developer can also confirm that his modeling is consistent by validating the graph. Any inconsistencies that are found are reported to him (e.g., through a graphical representation such as Figure 6.8). As stated previously, it is not possible to pinpoint the exact error, but only the cycle or triangle where there is an inconsistency. It is up to the user to review his analysis and correct the problem.

Figura 6.7: GMT Interface (MAC OSX)
6.4 Discussion

This section discusses some of the characteristics of the MCM language, namely, the choice of simultaneity and reachability as the dimensions for characterization, and presents a theoretical evaluation of the language.
6.4 Discussion

Figura 6.9: Requirements generation

6.4.1. MCW Characterization

The choice of reachability and simultaneity as the dimensions to characterize mobile work was discussed in Section 2.4. The use of reachability in a graph describing mobile work coincides with the definition of reachability for directed graph analysis. In this context, reachability means that one is able to get from one vertex to some other vertex. This possibility of communication and interaction is what this dimension aimed to capture: that two users will be able to communicate, even if they require an intermediary to do so.

MCM graphs contain information about reachability and simultaneity, as well as interactions and information flow. The information used to create them also contains working hours, places, devices, and networking information. However, it would be possible to add to the graphs more information, such as the proportion of time spent in each scenario (to give priority to the scenarios that are used the most) and the level of detail each role requires in the application (e.g., a super-
visor may need to be able to see all his employees’ information, but they should not be able to access the supervisor’s data). These improvements could make the language more expressive, however, they would also make the graphs difficult to understand and create. This tradeoff could be studied in more detail in future work.

The main limitation of the reachability/simultaneity approach as modeled here has to do with the definition of reachability. Reachability is defined as the lack of a communication channel between two users, or the unavailability of one of the users. However, unavailability is more complex than this: sometimes personal preferences, social circumstances, urgency of work, etc, may cause one person to be available for some people or tasks, and unavailable to others. This subtle difference is not captured in an MCM graph.

6.4.2. MCM Language theoretical evaluation

In [113], David Pinelle discusses requirements for analyzing loosely coupled work in preparation for groupware design. The requirements he proposes are the following:

- **Coupling patterns:** The analysis technique should capture the level of integration and interdependence between elements, incorporating information about all relevant relationships.

- **Work patterns:** The analysis technique should capture the workgroup’s work patterns. This requires an analysis of individual and collaborative tasks, along with the points of interdependence. Other relevant factors are work locations and devices that are used.

- **Communication, coordination, and information utilization patterns:** The analysis technique should capture the current collaboration patterns and the frequency, directness and richness of collaboration channels. Problems in communication and coordination should also be considered.

- **Causes and outcomes:** The analysis technique should capture the reasons for loose coupling and outcomes associated with the adoption of loose coupling.
This section discusses how MCM graphs address some of the proposed requirements. The goal of this is to see whether the MCM proposal aligns with this previously presented work. The requirements are accomplished in the following way:

- **Coupling patterns:** MCM graphs incorporate information about relationships among all collaborators. These graphs do not include stakeholders outside of the collaboration.

- **Work patterns:** MCM graphs do not display tasks, but they do show points of shared work. The characterization of roles associated to an MCM graph presents work locations and work artifacts.

- **Communication, coordination, and information utilization patterns:** MCM graphs may be used to capture current collaboration patterns. They allow developers to find potential improvements to the collaboration as well as breakdowns in the collaboration (e.g., when two roles are in unreachable and non-simultaneous scenarios).

During the software development process, requirements or work patterns may change. These changes should be reflected in the application that is being developed. For example, during development of an application for construction inspections, it may be decided that due to budget constraints, a synchronization server will be in the project leader’s personal computer, which is only available during the hours the project leader is in his office. This means that the collaboration scenario between the project leader and headquarters server is simultaneous and reachable. If we incorporate this change into an MCM graph and run the validation tool, an error may be detected that may cause a chain reaction of changes in the interaction scenarios of the whole group. MCM graphs may help detect these changes to make sure they are reflected in the application.
Capítulo 7

Experimental Results

This chapter describes the experiments that were made to validate the hypotheses. The modeling language and evaluation method were proposed after a literature review and an informal observation of firefighters conducting an urban search and rescue simulation. Afterwards, the validation was done examining other instances of loosely coupled mobile collaboration and contrasting the proposed methods with existing ones. The scenarios of validation were the following ones:

- **Urban Search and Rescue**: Urban search and rescue was studied more formally a second time, recording all radio communication and conducting interviews, to confirm that the initial observations were accurate.

- **Construction Inspections**: An application to support construction inspections was also modeled and evaluated using MCE evaluation. The goal of this study was to see how MCM modeling applied to a completely different setting.

- **Healthcare**: Researchers at CICESE \(^1\) have studied informal communication in hospital settings for several years. They have evaluated their solutions using focus groups. We evaluated their initial application with MCE evaluation, and contrast our findings with the results from their own evaluation.

- **Daily Emergency Management**: During the development of this thesis, an application to support everyday emergencies, called MobileMap, was im-

\(^1\)Centro de Investigación Científica y de Educación Superior de Ensenada BC, Mexico, http://www.cicese.mx/
implemented. This application had three iterations, so each application was evaluated using MCE evaluation. Then, focus groups were used to contrast MCE evaluation with opinions from users.

This chapter is organized as follows. First, an explanation of the experimental method is presented in more detail. Then, the four mentioned scenarios of validation are discussed. For each, the setting is described and MCM modeling or evaluation is applied. Then, the results from these experiments are discussed. Finally, the evaluation cost for MCE is analyzed.

7.1. Experimental Method

The goal of these experiments was to study hypothesis H1: An evaluation method applied during the development process can help predict how well a mobile shared workspace supports collaborative work in a specific work context. The experiment was the following. MCE was applied to three applications developed by independent teams for mobile collaborative work settings. Then, the errors found by our method were contrasted with the results from other evaluation methods. The experimental method is summarized in Table 7.1. This chapter discusses these three settings as well as the urban search and rescue scenario.

Cuadro 7.1: Experimental method

<table>
<thead>
<tr>
<th>Setting</th>
<th>First eval.</th>
<th>Second eval.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construction</td>
<td>MCE</td>
<td>Comparison</td>
</tr>
<tr>
<td>Healthcare</td>
<td>MCE</td>
<td>Focus Groups</td>
</tr>
<tr>
<td>Firefighters</td>
<td>MCE</td>
<td>Focus Groups</td>
</tr>
</tbody>
</table>

7.2. Urban Search and Rescue

Firefighters have expanded their traditional area of work (fire emergencies) to also include urban search and rescue operations (USAR). USAR are operations
7.2 Urban Search and Rescue

mounted by emergency services to reduce the consequences of a catastrophic event such as an earthquake, flood, or terrorist attack in an urban area. These operations differ from everyday firefighter work in that they may require a larger number of workers (sometimes from other areas, regions, or countries) and extend over a longer period of time. The goal of these operations is to rescue trapped victims while minimizing risks for victims and emergency workers.

The operational structure of firefighters working in emergency sites is shown in Figure 7.1 (based on [71] and [77], and confirmed by our interviews). The incident commander is in charge of the entire rescue effort. Several firefighters or teams of firefighters support his/her work, e.g. operations and logistics. The operations officer is in charge of coordinating the firefighters working in teams at the emergency site. It should be noted that firefighters are organized in a strictly hierarchical way.

Part of the difficulty of studying USAR stems from the fact large-scale emergencies are rare. Besides, it is usually not permitted to observe firefighters work in them. Therefore, to observe firefighters conducting search and rescue, we studied a USAR training course extending over three days and finishing with a simulation of an emergency, in which instructors create a setting that is as realistic as possible. We observed parts of two courses, including two complete simulations, as well as some shorter training exercises. Each simulation lasted for over two hours, and the total observation time was around seven hours. The observations are summarized here, and can be found in more detail in Appendix A.
7.2 Urban Search and Rescue

7.2.1. Validation Process

The second observed simulation consisted of a new group of firefighters simulating the aftermath of an earthquake and the search for victims. All radio communications and some video of the exercise were recorded during this simulation. This data was used to validate the MCM modeling language and to test the application of MCE. Afterwards, this was coded to study times, involved people, and purpose of messages. This data is discussed from the perspective of the MCM language, i.e., involved roles, simultaneity and reachability.

7.2.1.1. Roles and Communication

We observed 934 radio messages, 6 (0.65%) of which were explicitly sent to several recipients; the remaining ones were between one sender and one recipient, although any person with a radio could listen to them. These messages are displayed in Fig. 7.2, where the thickness of the edges represents communication frequency, and the numbers detail the number of recorded messages. This graph data is consistent with the idea of modeling communications as 1-1 messages and the importance of identifying roles. For example, in this case there are three actors who are team leaders (G1, G2 and G3) and their communication is very similar (few communications among them, and high communication - 152, 49, and 132 - with the OO).

7.2.1.2. Simultaneity

Firefighters in USAR simulations work simultaneously during the emergency. Fig. 7.3 displays the senders and recipients of messages, graphed versus time. We can observe all the users with radio either send or receive messages throughout the whole emergency. The OO is constantly organizing the operation through the radio, but all the actors have to pay attention to it, because each one is involved and may be contacted at any point during the entire duration of the emergency.
7.2 Urban Search and Rescue

Figura 7.2: Radio communications

Figura 7.3: Simultaneity
7.2.1.3. Reachability

Finally, we tried to graph an approximation of unreachability using the radio. This measure was taken by counting the number of times a user had to contact another one more than once to get a reply. Again, this is a conservative estimate, because we are not taking into account interruptions, times when a user was contacted and did not reply, times when a user was unavailable, etc. The directed graph in Fig. 7.4 displays the instances in which a user had to call another one more than once to be able to reach him or her. We can note more than 10\% of all messages are wasted trying to contact another user, therefore wasting radio time as well.

![Figure 7.4: Unreachability](image)

7.2.2. MCM Graph

The process of building the graph started by identifying the roles that could benefit from a pervasive collaborative system. Since the same firefighters using radio communication are the ones making decisions and organizing work, these are the identified roles. The IC and LO work from a fixed location, so they can use a device with greater capabilities (more processing power, larger screen). The rest of the users are highly mobile, so a smaller and lightweight device is more appropriate for them. USAR operations after an earthquake or similar disaster cannot depend on external infrastructure, so the network type is an ad-hoc network. Then, we continued by identifying the information flows between roles. Here, we can describe
which roles have bidirectional communication (e.g., OO both sends and receives information from SO) and which roles have unidirectional communication (e.g., GX report their location to IC, but they do not receive direct communication from him, as they are directed by OO). Comments could also be added to specify the type of information flow or interaction (e.g., team leaders’ devices should report their location to the IC, but they do not communicate directly). All of this information is summarized in Table 7.2.

Cuadro 7.2: Identified roles and characteristics for USAR operations, and existing information flows between them

<table>
<thead>
<tr>
<th>Role</th>
<th>Characterization</th>
<th>Information flows</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Location</td>
<td>Devices</td>
</tr>
<tr>
<td>IC</td>
<td>Support area</td>
<td>Netbook</td>
</tr>
<tr>
<td>LO</td>
<td>Support area</td>
<td>Netbook</td>
</tr>
<tr>
<td>OO</td>
<td>Emergency area</td>
<td>Smartphone</td>
</tr>
<tr>
<td>SO</td>
<td>Emergency area</td>
<td>Smartphone</td>
</tr>
<tr>
<td>GX</td>
<td>Emergency area</td>
<td>Smartphone</td>
</tr>
</tbody>
</table>

An MCM graph describing the collaborative process may be drawn with this information (Fig. 7.5). This graph represents the collaborative scenario of USAR operations. It identifies the roles, interactions between them, and the possible collaboration scenarios in which two actors may be when they require an interaction.

7.2.3. Discussion

The graph presented in Fig. 7.5 is simple, but it conveys important knowledge about the collaborative process:

- It identifies all the roles that will interact through the system, distinguishing which roles receive and/or send information.
- It also specifies communication between specific users. The hierarchical structure of firefighters means, e.g., IC will not communicate with GX directly, only through OO.
In this scenario, all users work simultaneously. This is because the system is intended for work during an emergency, from beginning to end.

At some point in time, some users (e.g., SO) may become unreachable. This may affect the design of the system, because mechanisms to provide connection flexibility, offline awareness, etc., should be implemented.

The graph and characterization provide important information that is essential to the design of the system, e.g., since users in the field may become unreachable, developers may choose to devise ways to minimize unreachability. In the literature, we can find several examples in which we can see that this is important knowledge about the system. For example, the Siren system [71] uses a peer-to-peer architecture, with store-and-forward communication. In this way, intermediary nodes are used to transfer information between firefighters in the field. Luyten et al. [82] use a different approach. They propose a mesh network that is dynamically built in the disaster area, and is connected to a wired or wireless network. These are both solutions that deal with unreachability caused by lack of a communication channel. Identifying issues such as these is one of the benefits of analyzing and modeling a collaborative mobile scenario.
7.3. Construction Inspections

Construction inspections are projects in which an extended area such as a building is examined by several companies specializing in technical inspection, usually of just one aspect of the construction (e.g. infrastructure, paint). Each company has one or several inspectors, who critically examine areas of the building which may overlap, and naturally overlap with the areas inspected by other companies. Work is highly mobile and workers collaborate in a loosely coupled way. Several challenges are present in supporting construction work, e.g. efficiently disseminating project information in the field, and providing access control and security [137].

In the case of this scenario, the existing application has not been formally evaluated, so there is no comparison between MCE and another evaluation method. In this case, we test how well the MCM language describes a new scenario and whether the MCE evaluation method can provide hints on features that have not been implemented in the software system yet. This was accomplished through generating the suggested requirements for the software and checking the application to see if they had been implemented. The current developer for this project was present during the evaluation to comment whether features were missing due to time constraints or because they had not been considered.

7.3.1. Description of work

Construction inspections typically involve a main contractor and several subcontracted companies, e.g. for electric installations or painting. Each company has a team of workers who inspect the construction to identify contingency issues. Each team has a chief inspector in charge of organizing the work of several inspectors. The task of finding problems in an area of the building may be split into subtasks for each team member, who may be inspecting a subarea or inspecting for a specific type of problem. In either case, the chief inspector must combine the inspectors’ reports into one and report it to the contractor. Most construction
projects also have a foreman on-site, who also should be notified of the inspection results.

7.3.2. Software

An application to support this scenario was developed for tablet PCs [108]. Its interface is shown in Figure 7.6. This version heavily focuses on providing collaborative capabilities for construction inspection workers, which has been informally evaluated by inspectors as the most interesting and novel feature of the software.

Figura 7.6: ITO interface (a) General view, and (b) detail of the central area [127].

The application, called ITO, allows inspectors to conduct the reviewing process and also to interact among them and with the foreman/main contractor. The main characteristics the software intends to support are the following [128]:

- Several inspectors distributed in an extended area, jointly working on diagnosing the construction status.
- Inspectors need to know other inspectors’ locations and current activities to collaborate on-demand.
- Users are not always able to use a fixed communication infrastructure.
- Support for sharing data and interacting is required. However, workers cannot know where and when the interactions will take place.
7.3.3. Discussion

The authors of the application used the MCM language to model the collaborative scenario of construction inspections. This graph is shown in Figure 7.7. With this graph, we automatically generated the requirements to find whether any suggested requirements had been overlooked in the software. The details of this evaluation are presented in Appendix A. The obtained results are discussed in the next section.

![MCM Graph for construction inspections scenario](image)

There were several interesting results from the evaluation to the ITO application for construction inspections. The development of this application is currently heavily focused on supporting collaborative work, so the results from the evaluation process were fed back into development to improve collaborative support. The suggestions made by MCE were generally regarded as important and reasonable.

The evaluation uncovered two features which are planned but currently not implemented, which are the following:

- **Ad-hoc work sessions:** Currently, there are no security measures in the software. A developer is working on encrypting communications between the devices to protect information. However, any device containing a copy of the
application may participate in the inspection process, which may compromise the security of the evaluation process.

- **Explicit data replication:** The application does not implement this feature, although there is an external module that can be used to copy data. Developers are working on integrating the module into the application.

The evaluation also found several requirements that can be added to the software and were not currently considered. These requirements were found by the developer to be important additions to the software. They are the following ones:

- **Asynchronous messaging:** Some users may work in shifts. Therefore, asynchronous messaging can be implemented to deliver messages to users who are not working simultaneously.

- **Pushing notifications:** Construction inspections have connection flexibility, i.e., it is expected that users will connect and disconnect frequently. Pushing notifications could help notify a user when another has become reachable.

- **Offline awareness:** The application currently only implements online awareness. For example, the application displays the estimated location only of reachable users. This information could also be shown when the user is unreachable, accompanied e.g. with the elapsed time since the last connection. Also, currently the list of users only presents reachable devices, which could be improved to also display users who have been reachable in previous moments.

- **User gossip:** The application provides message routing, and predefined messages such as “Let’s meet to discuss the inspection”. However, unreachable users cannot receive messages. User gossip can be used to propagate a message such as the previous one to unreachable users.

The results of this evaluation are encouraging, since MCE detected several problems, which were either incomplete implementations or missing collaborative support. The application, however, did implement several of the suggested functional requirements, which confirms the fact these are important aspects of collaborative work that should be present in mobile collaborative applications.
7.4 Healthcare

Hospitals are settings in which teams composed of physicians, medical interns, nurses and other staff work to solve medical emergencies and take care of hospitalized patients. Hospital work is characterized by the need to coordinate specialists, task switching, an intense information exchange, data integration from many devices or artifacts, and the mobility of hospital staff, patients, documents and equipment [19, 94]. At the same time, this environment is highly critical and error prone. In fact, it has been estimated that up to 98,000 people die each year in US hospitals as a result of medical errors that could have been prevented [75].

7.4.1. Description of work

A study observing informal communication and collaboration in hospitals was recently reported [87, 94]. Researchers shadowed ten physicians for two work shifts, capturing all the interactions between them. Later on, this information was coded and analyzed. An interesting result from this study is that hospital workers spend 53% of their time outside their base location [94], i.e., they are highly mobile. While on the move, other staff often contacts them to request information or to discuss patient-related issues.

Moreover, face-to-face is not the only communication channel workers use to collaborate. In some situations, when one of the collaborators is unavailable (e.g., if they work during different shifts), medical workers use asynchronous communication by leaving instructions in information artifacts such as medical records.

We also conducted an interview and two focus groups to learn about unreachability in hospitals. In these focus groups, several hospital workers remarked that the problems of unavailability and locating workers are important and frequent in hospital work.
7.4.2. Software Evolution

A pervasive tool to capture collaborative interactions called *cTracker* was designed and developed by researchers in Mexico based on the previously mentioned study [87]. This application estimates the location of workers in a hospital, detects when an interaction between two or more users is occurring, and captures verbal and inscribed information. Interactions are then stored, along with contextual information such as the location where they were captured, time, topics, and artifacts that were used and generated. When a user needs to recall some information or find a file that was modified during an interaction, he can search for it filtering by location, participant, time of occurrence, etc. A screen shot of the application is shown in Fig 7.8.

![cTracker interface](image)

Figura 7.8: cTracker interface

The cTracker application was evaluated through a focus group with several hospital workers, who commented if the identified problems were frequent and whether they would use the application. Unfortunately, we were not able to obtain the data from the evaluation, but the researchers privately shared with us the features that were added to cTracker after the evaluation [42], which we now describe.
7.4 Healthcare

The general results of the evaluation were good, and users seemed interested in using the software, even though several expressed concern with privacy issues. The suggestions the researchers obtained during the focus group suggested a need for communicating with users even when they were not nearby or interacting directly. The second version of the tool, modified as a result of the evaluation, includes a module that allows users to leave voice messages to unavailable users. This is accomplished through interfaces for message recording, sending, and reviewing received messages. The goal of this new module is to be able to interact asynchronously with users even when they are busy, or work in another shift. Even though we could not access the data from this evaluation, we compare the changes made by the designers to the system with the changes that would have been suggested by MCE.

7.4.3. MCE Evaluation

The first step towards building the MCM graph was identifying the roles to be played by system users. Physicians, specialists, medical interns and nurses were the roles we found from the cited study (and confirmed by our focus group). We differentiate between the physicians, specialists and interns because hospital work is somewhat hierarchical. In particular, specialists are reluctant to be located by interns, whereas interns accept being located as part of their work. The application was developed for a PDA or smartphone due to the high mobility hospital workers experience, and the network access in a hospital is a wireless network. Hospital workers work in shifts, since the hospital has to remain open at all times. Each role is characterized in Table 7.3.

We then described the information flows between roles. These are shown in Table 7.4. In this case, we may make nurses and interns only able to exchange information among them and provide information to physicians and specialists, to acknowledge their request for privacy. We then built an MCM graph describing the collaborative process in hospital work using the gathered information (Fig. 7.9).
Cuadro 7.3: Identified roles and characteristics for hospital settings

<table>
<thead>
<tr>
<th>Role</th>
<th>Location</th>
<th>Devices</th>
<th>Working hours</th>
<th>Network access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physician</td>
<td>Hospital</td>
<td>Smartphone</td>
<td>Work in shifts</td>
<td>Wireless network</td>
</tr>
<tr>
<td>Specialist</td>
<td>Hospital</td>
<td>Smartphone</td>
<td>Work in shifts</td>
<td>Wireless network</td>
</tr>
<tr>
<td>Nurse</td>
<td>Hospital</td>
<td>Smartphone</td>
<td>Work in shifts</td>
<td>Wireless network</td>
</tr>
<tr>
<td>Intern</td>
<td>Hospital</td>
<td>Smartphone</td>
<td>Work in shifts</td>
<td>Wireless network</td>
</tr>
</tbody>
</table>

Cuadro 7.4: Roles and relationships

<table>
<thead>
<tr>
<th>Role</th>
<th>Physician</th>
<th>Specialist</th>
<th>Nurse</th>
<th>Intern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physician</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Specialist</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nurse</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Intern</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>

There is relevant information in the graph for hospital work presented in Figure 7.9. At first glance, it is clear this graph shows work is done in shifts in this setting. However, the first version of cTracker did not take this information into account. We can see that using an analysis technique can help make requirements present to designers. We can make the following observations:

- All users work non-simultaneously at some point in this scenario. This is due to the fact they work in shifts, so e.g. a nurse leaving her shift will probably require asynchronous communication with the nurse beginning the next shift. This is currently done through medical records, but a technology-based system for hospital work may include support for this situation.

- Users become unreachable in a hospital because they may be unavailable (e.g., busy with an emergency surgery). In contrast, firefighters doing USAR operations usually become unreachable because they lack a communication channel (even the radio, which they currently use, is frequently unheard).

We can automatically generate the requirements for this scenario. They are detailed in Table 7.5. Since the interactions of all users in this case are the same, we display the requirements for one generic user called Staff. In this case, the first
column displays the requirements. The second column has marks on the requirements that were automatically generated by the MCE evaluation, and the third column displays the requirements that, according to the scenario and evaluator judgement, are actually required. Then, both versions of cTracker are compared. The requirements that were implemented are marked in green, the requirements that were not implemented are in red, and we marked in yellow the ones in which not enough information is available to distinguish whether they were implemented or not.

![MCM Graph for hospital setting](image)

**Figura 7.9: MCM Graph for hospital setting**

### 7.4.4. Discussion

Table 7.6 displays a summary of the requirements that were implemented. We can see that the first version of cTracker, which neglected to include support for users who at times were unavailable, has a success rate of 40%, while the second version has 60%. If we exclude the requirements with an unknown status, the
success rates are 54.5% and 81.8% respectively. This information leads us to believe that, had the developers of cTracker used the MCE evaluation method, the requirements for this particular scenario would have been clearer from the start, and results of the first focus group would have been better. We can see that the evolution of the software matches the improvement in the MCE evaluation results, which is to be expected, since new collaborative features were added in the second iteration.

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Staff</th>
<th>Generated?</th>
<th>Required?</th>
<th>cTracker v1</th>
<th>cTracker v2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-hoc work sessions</td>
<td>X</td>
<td>X</td>
<td>Unknown</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>Asynchronous messaging</td>
<td>X</td>
<td>X</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Automatic connection</td>
<td>X</td>
<td>X</td>
<td>Unknown</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>Automatic peer detection</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Caching</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>X</td>
<td>X</td>
<td>Unknown</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>Explicit data replication</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>File transfer</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Message routing</td>
<td>X</td>
<td>X</td>
<td>Unknown</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>Pushing notifications</td>
<td>X</td>
<td>X</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Offline awareness</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Online awareness</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Synchronous messaging</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Transition awareness</td>
<td>X</td>
<td>X</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User connection/disconnection</td>
<td>X</td>
<td>X</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>User gossip</td>
<td>X</td>
<td>X</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Cuadro 7.6: Results from MCE evaluation to cTracker v1 and v2

<table>
<thead>
<tr>
<th>Statistics</th>
<th>cTracker v1</th>
<th>cTracker v2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generated</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Required</td>
<td>15 (93,8%)</td>
<td>15 (93,8%)</td>
</tr>
<tr>
<td>Implemented</td>
<td>6/15 (40%)</td>
<td>9/15 (60%)</td>
</tr>
<tr>
<td>Not implemented</td>
<td>5/15 (33,3%)</td>
<td>2/15 (13,3%)</td>
</tr>
<tr>
<td>Unknown</td>
<td>4/15 (26,7%)</td>
<td>4/15 (26,7%)</td>
</tr>
</tbody>
</table>
Naturally, an interesting question concerns the two requirements that were generated by MCE and were not implemented in the second version of cTracker. There are several possible explanations for this situation: the two missing features may have been suggested by users and not considered important by the development team for this version of cTracker, or they may not have been suggested by users, or they may not actually be important in this scenario and were incorrectly judged to be required. However, the most probable explanation is the users might have not realized the importance of the two missing requirements (transitions between connected and disconnected states, and of finding other users), because the evaluation was a focus group in which the software features were presented but there was no time for actual experimenting with the software.

7.5. Daily emergency management

Firefighters are in charge of dealing with several types of emergencies threatening civilians and property, such as fires, car accidents, forest fires, hazardous materials, and natural disasters such as earthquakes. Firefighters in Chile, as in many Latin American countries, are volunteers. This means they do not receive a salary for their job, and they have minimal support from government agencies.

This section describes the work of firefighters in everyday emergencies, e.g. fires, car crashes, explosions, building collapse, etc. This information was gathered from interviews, visits to a command center, and existing literature.

7.5.1. Description of work

The emergency response process begins when the command center, a special unit in charge of coordinating efforts done by several fire companies, receives a phone call (typically from one or several witnesses) indicating an emergency situation. The command center dispatches one or more fire trucks depending on the emergency site, the event size and type, and the available resources. For this purpose, the command center must notify the involved companies. These notifications
have been done by several media in the past, depending on available technology. In colonial times, church bells were used; later on, it was a siren. Today, radio devices are commonly used.

Typically, each command center coordinates the activities of 10-15 fire companies. Each company has a headquarters and between 2 and 4 fire trucks which can be sent to various emergencies. Usually, the firefighting community composed of a command center and its depending fire companies count on 2 or 3 radio channels to coordinate the response efforts to emergency situations. In the case of urban areas, there can be up to 15 simultaneous emergencies that need to be coordinated by the command center.

Response time is critical in urban response processes [47, 88, 107]. However, fire trucks currently depend on the drivers’ skills to arrive to the emergency site. Once there, firefighters ask the command center for additional information, such as hydrants or the location of other fire trucks. This radio-based information exchange continues during the whole response process. Frequently, the command center needs to communicate with other emergency organizations (e.g. police departments, health services or hospitals) in order to coordinate their efforts. The level of coordination among these organizations has direct impact on the number of saved lives and the magnitude of the affected civil infrastructure [97, 107].

Although radio-based communication has several benefits, it is currently a limitation for the emergency response processes, particularly in urban areas. In these cases it may be important for firefighters to have information about electrical/gas networks, building blueprints or the types of physical infrastructure around the affected area (e.g. schools or elderly-care houses). This information can be relevant to organize and conduct the response process, but unfortunately it is almost impossible to transmit through a radio channel.

Furthermore, it has been widely recognized there is an insufficient number of channels to coordinate urban emergencies [2, 97, 107]. One cause of this shortage is the number of simultaneous emergencies that typically occurs in urban areas.
Another cause is the number of radio transmitters that are able to transmit/receive messages. Currently, there is almost one radio-transmitter per firefighter. Some of the problems caused by this situation are the following:

1. Firefighters have to improvise during a response process because they are not able to ask for additional information [2, 88].

2. Due to the scarce number of communication channels, a message delivered by an equipment with high spread power overwrites any other message distributed by the channel during the same time interval [84, 89].

3. Relevant information about the affected area is available at the control center, but it cannot be sent fast enough to the firefighter because it is hard-copied or in digital formats [2, 107].

4. Messages do not have priority; therefore, important messages (e.g., an alarm indicating the possible collapse of a building during a fire) could fail to be received by first responders.[24, 84].

7.5.2. MCE Evaluation

MobileMap is a mobile collaborative Graphical Information System (GIS) designed for use in everyday emergencies that firefighters face [92, 93]. The main ideas behind this application were to provide a low-cost solution to reduce radio use and improve both the time firefighters take to arrive at an emergency site and the knowledge about the emergency when they arrive to it. The MobileMap application was developed in three stages, from 2007 to 2009. We now briefly describe the three iterations of the application (the third iteration has two software versions, 3.0 and 3.1) and apply MCE evaluation to each.

7.5.2.1. Version 1.0

Description

The first version of MobileMap was created by an engineering student as his diploma thesis and finished around January 2007 [147]. The goal of this work was
to create a PDA application to allow workers in the field to manage cartographical information to support the decision making process. The interface of the resulting application is shown in Fig. 7.10.

![Figura 7.10: MobileMap v1.0 interface](image)

The collaborative capabilities in this version were few. Only one type of user was considered, and he could exchange information with other users in emergency situations. For example, a firefighter in charge of search and rescue could exchange information with a civil engineer in charge of researching the stability of the affected civil infrastructure. When the two actors were near, they could synchronize their information, i.e., exchange the files containing points of interest.

Data exchange was as follows (Fig. 7.11): a user had to input the IP address of the device he wanted to communicate with. Then, the user obtained the list of points of interest that his collaborator had defined. He could choose the files he wanted to obtain from the remote application. When he had obtained the information, it could be visualized in the map.
### Evaluation

We now describe the MCE evaluation process for the first version of the MobileMap application.

**Step G.1. Identification of roles**

The developer of this first version only identified one type of role, which we will call *Firefighter*. This generic user, who could be a firefighter chief, engineer chief, etc., would use the application to add, view and share points of interest in a map.

**Step G.2. Role characterization**

The location of work was generally around the emergency area. The application was designed for PDAs with Windows Mobile OS. The users would work during the emergency and the network access was an ad-hoc network formed between the two interacting devices. The characterization of this role is shown in Table 7.7.

**Step G.3. Identification of relationships**

Next, we identify the relationships. This is very simple in the case of this application because we must only examine one possible relationship, which is between two firefighters. The result is that firefighters may interact between them to
Cuadro 7.7: Identified roles and characteristics for MobileMap v1.0

<table>
<thead>
<tr>
<th>Role</th>
<th>Location</th>
<th>Devices</th>
<th>Working hours</th>
<th>Network access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefighter</td>
<td>Emergency area</td>
<td>PDA</td>
<td>During emergency</td>
<td>Ad-hoc</td>
</tr>
</tbody>
</table>

synchronize information, through the previously described mechanism. Table 7.8 describes this interaction.

Cuadro 7.8: Roles and relationships

<table>
<thead>
<tr>
<th>Role</th>
<th>Firefighter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefighter</td>
<td>Yes; bidirectional information synchronization</td>
</tr>
</tbody>
</table>

*Steps G.4-G.5 and E.1-E.4*

We can now generate the MCM graph for the first version of the MobileMap application (Fig. 7.12). Naturally, the graph is also very simple, as it has only one identified type of user, interacting simultaneously and either reachable or unreachable. This graph allows us to automatically generate the requirements for this application, which are displayed in Table 7.9. This table displays the requirements for the application and an indication about the ones which were implemented at this point. We can see that the only implemented requirements were related to data transfer between two devices, which was actually the only collaborative capability of the application.

Figura 7.12: MCM Graph for MobileMap version 1.0
7.5 Daily emergency management

Cuadro 7.9: Requirements for MobileMap v1.0

<table>
<thead>
<tr>
<th>Requirements: Firefighter-Firefighter</th>
<th>Required?</th>
<th>Done?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-hoc work sessions</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Asynchronous messaging</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Automatic connection</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Automatic peer detection</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Caching</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Explicit data replication</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>File transfer</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Message routing</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Pushing notifications</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Offline awareness</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Online awareness</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Synchronous messaging</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Transition awareness</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>User connection/disconnection</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>User gossip</td>
<td>X</td>
<td>No</td>
</tr>
</tbody>
</table>

7.5.2.2. Version 2.0

Description

When the first version of the software was finished, new functionality was added by a group of engineering students as part of their software engineering coursework. The idea was to improve the application, allowing information sharing between mobile devices, improving usability, and incorporating the use of the GPS functionality of the PDA. Therefore, the new functionality was intended to improve the interface and application-specific features. The resulting application’s interface is presented in Figure 7.13.

Evaluation

Next, we describe the evaluation process for the second iteration in the development of MobileMap. In this case, steps 1, 2 and 3 of the MCE evaluation are the same as before, since there is still only one identified actor. Therefore, the generated requirements are also the same. However, when we evaluate whether the requirements were implemented in this iteration, there are several surprising results. The results of this evaluation are presented in Table 7.10. We can observe
Figura 7.13: MobileMap v2.0 interface

the following:

- **Automatic peer detection** and **automatic connection** are marked in yellow, because the requirements document identified these as requirements, but they were not implemented in the final version of the software. We can hypothesize that this was due to the complexity of these requirements, or that other requirements were prioritized and not enough time was allocated to implement them. This reinforces the idea that it is important to identify requirements early on and give importance to collaborative capabilities.

- The **File transfer** functionality, which was implemented in MobileMap version 1.0, was lost in this version. This was because developers did not remember to include and update the file transfer functionality to keep working in the new interface. The evaluation method highlights this mistake, as the results from the evaluation are actually worse than in the first iteration.

- It is important to note that there were improvements in this second version of MobileMap. For example, GPS functionality was incorporated, as was loading high amounts of data, and the interface was improved. This is not reflected in the evaluation results, because MCE evaluation focuses only on collaborative functionalities, not on usability or application-specific requirements. The collaborative capabilities in this case diminished, and so did the evaluation results.
7.5 Daily emergency management

Cuadro 7.10: Requirements for MobileMap v2.0

<table>
<thead>
<tr>
<th>Requirements: Firefighter-Firefighter</th>
<th>Required?</th>
<th>Done?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-hoc work sessions</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Asynchronous messaging</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Automatic connection</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Automatic peer detection</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Caching</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Explicit data replication</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>File transfer</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Message routing</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Pushing notifications</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Offline awareness</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Online awareness</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Synchronous messaging</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Transition awareness</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>User connection/disconnection</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>User gossip</td>
<td>X</td>
<td>No</td>
</tr>
</tbody>
</table>

7.5.2.3. Version 3.0

Description

After the second prototype was implemented, an engineering student constructed a third version for his diploma thesis [91]. This version was finished in March 2009. It added some collaborative features, it considered additional actors and implemented a central server with which users can synchronize information. This version uses a service-oriented architecture, where each handheld device and command center is a potential service consumer/provider. The devices interact using message passing through Wi-Fi or GSM networks. It is possible to identify three types of nodes in this scenario:

- **Command centers**, which do not report their location or their list of resources. These nodes expose a set of services allowing the described operation. For example, services to inform/retrieve the trucks’ location and the list of tools they contain, inform the current emergency situations and the trucks’ assignments. Other emergency management services are provided by information systems running in the command center.

- **Fire trucks**. By default, these nodes report their location and list of materials to the command center. They also expose a service that provides the same
information to other fire trucks or firefighters. Firemen can take advantage of this functionality when they are co-located in the emergency site using a mobile ad hoc network.

- **Firefighters.** These nodes can only consume information from command centers or fire trucks, and exchange file and messages using the mailbox.

Fig. 7.14 shows the interface of this version of MobileMap. Map navigation is possible by clicking on some point on the map. MobileMap will consider that point as a gravity point, therefore producing a shift on the map in the direction indicated by the user. Thus, it is easy and fast to navigate the maps. The navigation mode is the default mode of this tool; however MobileMap has other user modes which allow to do further operations on the maps. These modes are: navigation, destination, distance, information, current location, and fire trucks.

![MapaMovil interface](image)

**Figura 7.14: MobileMap v3.0 interface**

The *navigation* service allows firefighters to navigate the maps in two ways: on-demand or automatically. On-demand navigation requires the user to click on the screen with the stylus in order to produce a shift on the map. Automatic navigation is available for devices embedding GPS. It focuses on the user’s current location and automatically shifts the map according to the person’s movement. The GPS is also used for the *current location* mode, which allows users to center
the map on their current location.

The destination service allows a user to select a destination using an address or a tuple: latitude, longitude. Once a destination point has been selected, the application shows two arrows in the user’s current location. A blue arrow indicates the direction in which the user is moving, and a red arrow shows the direction in which the user must move to get to the destination point.

The information management functionality allows a firefighter to perform three operations: review emergency information, show information layers on the map, and exchange information with the command center and other firemen. Figure 7.15 shows the three forms displaying information about the emergencies. Fig. 7.15(a) shows a list of fire truck codes, the truck status code (e.g. 0-9: available), and additional data about each vehicle. If the user chooses the emergency option, a list of current emergency situations is shown on the screen (Figure 7.15(b)). It is possible to access the details of an emergency by selecting it (Fig. 7.15(c)). All this information is stored in the command center, and it may be selectively retrieved from the handheld devices.

On the other hand, if the user selects the views option s/he can choose the pre-loaded information to be deployed on the map, e.g. police offices or hospitals (Fig. 7.16(a)). Figure 7.16(b) presents the resulting map. It is possible to access
the institution information, e.g., its phone numbers or its working hours, by clicking on the corresponding icon.

Figura 7.16: Information deployment and exchange

The mailbox option (Fig. 7.16(b)) allows a user to access a shared folder which allows exchanging information with another fireman or with the command center. The information exchange can be done through Wi-Fi or GSM networks. Data exchange with the command center is typically done using GSM because usually there is no access to Wi-Fi networks. However, data exchange among firefighters located in the same emergency site is normally done using a Wi-Fi-based Mobile Ad hoc Network (MANET). Figure 7.16(c) shows the list of files a handheld device shares with the command center and other peer devices. The maps of the affected area or any other data can be shared in an easy and fast way through this mechanism.

Finally, the firetrucks option allows deploying on the map the current location of fire trucks, in a similar manner to the information management option. However, the information concerning vehicle location is dynamic, so the handheld reloads it after a configurable time period. The fire trucks’ position is obtained based on the location of the handheld belonging to each vehicle. Each handheld informs its position to the command center once the current position has changed in at least one hundred meters (in any direction) with respect to the previously reported location. Thus, parked trucks do not need more than one message to report their
7.5 Daily emergency management

position. Firefighters and other trucks selecting the fire truck option on MobileMap retrieve data on trucks from the command center. Knowing the trucks’ location is important for the response process, not only for the command center, but also for other firemen. Sometimes the emergency response process improves because a particular tool (belonging to a certain fire truck) has arrived to the emergency site.

Evaluation

Next, we describe the evaluation process for the third iteration of the MobileMap application.

Step G.1. Identification of roles

First, all the roles that are present in the collaboration are identified through direct observation. Besides the firefighter user identified in the previous iterations, we now incorporate the command center operator and a user in the fire truck. Also, a server was implemented in the command center. This server was used to capture the information input by the operators and reported by the PDAs, and to share it with other users.

Step G.2. Role characterization

Second, each role is characterized, describing location, devices used, working hours and network access. In this scenario, all mobile roles will use PDAs. The command center works on desktop computers. All roles (except for the dispatch center which is always active) work from the start of the emergency until it is resolved, so work is mostly simultaneous. The work location is the area of the emergency and the area used for organizing the rescue efforts, which is about 100 mts. away from the emergency area. The command center may be further away, but generally in the same city where the emergency is taking place. The role characterization is shown in Table 7.11.

Step G.3. Identification of relationships
7.5 Daily emergency management

Cuadro 7.11: Identified roles and characteristics for MobileMap v3.0

<table>
<thead>
<tr>
<th>Role</th>
<th>Location</th>
<th>Devices</th>
<th>Working hours</th>
<th>Network access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefighter</td>
<td>Any</td>
<td>PDA</td>
<td>Any</td>
<td>GPRS, MANET</td>
</tr>
<tr>
<td>Operator</td>
<td>Alarms Ctr</td>
<td>PC</td>
<td>Continuous</td>
<td>Internet</td>
</tr>
<tr>
<td>Fire Truck</td>
<td>Any</td>
<td>PDA</td>
<td>During Driving</td>
<td>GPRS</td>
</tr>
<tr>
<td>Server</td>
<td>Alarms Ctr</td>
<td>Server</td>
<td>Continuous</td>
<td>Internet</td>
</tr>
</tbody>
</table>

Next, we identify data flows between roles. These are described in Table 7.12.

Cuadro 7.12: Roles and relationships for MobileMap v3.0

<table>
<thead>
<tr>
<th>Role</th>
<th>Firefighter</th>
<th>Operator</th>
<th>Fire Truck</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefighter</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Operator</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>Fire Truck</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Server</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>

Steps G.4-G.5 and E.1-E.4

Finally, this characterization allowed us to create the MCM graph for the third version of MobileMap (Fig. 7.17). We then generated the requirements, which in this case are more complex, since we have six distinct relationships. Tables 7.13, 7.14 and 7.15 display the requirements and their level of accomplishment.

7.5.2.4. Version 3.1

Description

The third version of MobileMap had a final and small update when the development team decided a change in the communication network type. All communications would be through GSM networks and MANETs would not be used due to decreasing costs of mobile internet access and the fact that MobileMap supports small emergencies in which a loss of communications infrastructure is unlikely. We now analyze how this small change affects the MCM graph and evaluation.
7.5 Daily emergency management

Figura 7.17: MCM Graph for MobileMap version 3.0

Cuadro 7.13: Requirements for MobileMap v3.0 - Table 1

<table>
<thead>
<tr>
<th>Requirements</th>
<th>CC Operator</th>
<th>Firetruck</th>
<th>Firefighter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-hoc work sessions</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Asynchronous messaging</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Automatic connection</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Automatic peer detection</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Caching</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Explicit data replication</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>File transfer</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Message routing</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Pushing notifications</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Offline awareness</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Online awareness</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Synchronous messaging</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Transition awareness</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>User connection/disconnection</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>User gossip</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
### Cuadro 7.14: Requirements for MobileMap v3.0 - Table 2

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Firetruck Req?</th>
<th>Firetruck Done?</th>
<th>Firefighter Req?</th>
<th>Firefighter Done?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-hoc work sessions</td>
<td>X</td>
<td></td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Asynchronous messaging</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Automatic connection</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Automatic peer detection</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Caching</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Explicit data replication</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>File transfer</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Message routing</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pushing notifications</td>
<td>X</td>
<td>No</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Offline awareness</td>
<td>X</td>
<td>No</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Online awareness</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Synchronous messaging</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Transition awareness</td>
<td>X</td>
<td>No</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>User connection/disconnection</td>
<td>X</td>
<td>No</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>User gossip</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

### Cuadro 7.15: Requirements for MobileMap v3.0 - Table 3

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Firefighter Required?</th>
<th>Firefighter Done?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-hoc work sessions</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Asynchronous messaging</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Automatic connection</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Automatic peer detection</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Caching</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Explicit data replication</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>File transfer</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Message routing</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Pushing notifications</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Offline awareness</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Online awareness</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Synchronous messaging</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Transition awareness</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>User connection/disconnection</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>User gossip</td>
<td>X</td>
<td>No</td>
</tr>
</tbody>
</table>
Evaluation

The evaluation in this case is very similar to the previous evaluation, so we only note the changes introduced by the modification in the network type.

**Step G.2. Role characterization**

The firefighters are now working through GPRS (or GSM networks), and MANET networks are not used. Therefore, the new role characterization is shown in Table 7.16.

Cuadro 7.16: Identified roles and characteristics for MobileMap v3.1

<table>
<thead>
<tr>
<th>Role</th>
<th>Location</th>
<th>Devices</th>
<th>Working hours</th>
<th>Network access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firefighter</td>
<td>Any</td>
<td>PDA</td>
<td>Any</td>
<td>GPRS</td>
</tr>
<tr>
<td>Operator</td>
<td>Alarms Ctr</td>
<td>PC</td>
<td>Continuous</td>
<td>Internet</td>
</tr>
<tr>
<td>Fire Truck</td>
<td>Any</td>
<td>PDA</td>
<td>During Driving</td>
<td>GPRS</td>
</tr>
<tr>
<td>Server</td>
<td>Alarms Ctr</td>
<td>Server</td>
<td>Continuous</td>
<td>Internet</td>
</tr>
</tbody>
</table>

**Steps G.4-G.5 and E.1-E.4**

The change in the characterization of firefighters affects their reachability. Finally, this characterization allowed us to create the MCM graph for the third version of MobileMap (Fig. 7.18). We then generated the requirements, which in this case are complex, since we have six distinct relationships. Tables 7.17, 7.18 and 7.19 display the requirements and their level of accomplishment.

### 7.5.3. Focus Group Evaluation

MobileMap was evaluated using three focus groups. Each focus group began with a small introduction about the project and the consent of participants to audio record the focus group. Then, there was a brief powerpoint presentation about the goals and main features of the software. Finally, there was a group discussion: the participants were free to discuss software features, and they were also asked several questions to guide the discussion, prompting comments and criticism from
### Cuadro 7.17: Requirements for MobileMap v3.1 - Table 1

<table>
<thead>
<tr>
<th>Requirements</th>
<th>CC Operator</th>
<th>Firetruck</th>
<th>Firefighter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-hoc work sessions</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Asynchronous messaging</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Automatic connection</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Automatic peer detection</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Caching</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Explicit data replication</td>
<td>X</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>File transfer</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Message routing</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Pushing notifications</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Offline awareness</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Online awareness</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Synchronous messaging</td>
<td>X</td>
<td>Yes</td>
<td>X</td>
</tr>
<tr>
<td>Transition awareness</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>User connection/disconnection</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>User gossip</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Cuadro 7.18: Requirements for MobileMap v3.1 - Table 2

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Firetruck</th>
<th>Firefighter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Req?</td>
<td>Done?</td>
</tr>
<tr>
<td>Ad-hoc work sessions</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Asynchronous messaging</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Automatic connection</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Automatic peer detection</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Caching</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Explicit data replication</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>File transfer</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Message routing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pushing notifications</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Offline awareness</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>Online awareness</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Synchronous messaging</td>
<td>X</td>
<td>Yes</td>
</tr>
<tr>
<td>Transition awareness</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>User connection/disconnection</td>
<td>X</td>
<td>No</td>
</tr>
<tr>
<td>User gossip</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Cuadro 7.19: Requirements for MobileMap v3.1 - Table 3

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Firefighter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Required?</td>
</tr>
<tr>
<td>Ad-hoc work sessions</td>
<td>X</td>
</tr>
<tr>
<td>Asynchronous messaging</td>
<td>X</td>
</tr>
<tr>
<td>Automatic connection</td>
<td>X</td>
</tr>
<tr>
<td>Automatic peer detection</td>
<td>X</td>
</tr>
<tr>
<td>Caching</td>
<td>X</td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>X</td>
</tr>
<tr>
<td>Explicit data replication</td>
<td>X</td>
</tr>
<tr>
<td>File transfer</td>
<td>X</td>
</tr>
<tr>
<td>Message routing</td>
<td></td>
</tr>
<tr>
<td>Pushing notifications</td>
<td>X</td>
</tr>
<tr>
<td>Offline awareness</td>
<td>X</td>
</tr>
<tr>
<td>Online awareness</td>
<td>X</td>
</tr>
<tr>
<td>Synchronous messaging</td>
<td>X</td>
</tr>
<tr>
<td>Transition awareness</td>
<td>X</td>
</tr>
<tr>
<td>User connection/disconnection</td>
<td>X</td>
</tr>
<tr>
<td>User gossip</td>
<td>X</td>
</tr>
</tbody>
</table>
them. There were three researchers per focus group: a facilitator, a presenter, and a helper. I participated as the presenter, showing screenshots and explaining software features, and after the presentation simply recorded the discussion, while the facilitator guided discussion and asked questions. The helper was an engineering student who also was a firefighter, and he was inside the room to help with any unexpected requirements. The results of MCE were not available at the time of the focus group, however, the facilitator did try to guide some of the discussion towards interaction between firefighters, to gain insight into the required collaboration support. It is important to note that users were not explicitly asked to comment on the list of requirements; rather, they participated in a free discussion with a small amount of guidance towards collaborative aspects.

Each focus group lasted around an hour and fifteen minutes. Table 7.20 describes each focus group, detailing the number of participants, the number of represented fire companies, and the number of participants with leadership positions (e.g. commanders, captains, communication officers, etc.).

### Cuadro 7.20: Focus group participants

<table>
<thead>
<tr>
<th>Focus Group</th>
<th>Participants</th>
<th>Male</th>
<th>Female</th>
<th>Companies</th>
<th>Leaders</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

#### 7.5.3.1. First focus group

The results of the first focus group were positive. Participants were enthusiastic and all of them expressed interest in using the application by the end of the session. This section discusses the main themes that were discussed and the criticism and suggestions made by the participants.
7.5 Daily emergency management

Information availability

The focus group participants approved the information that is currently available in MobileMap, but they requested additional information to help them prepare for arrival at an emergency and help other firefighters arriving later understand the scenario. One firefighter commented that “having this information preloaded is the best we could have, truly useful”.

- **Building information.** When firefighters arrive at a building, they find the doorman is “shocked, and concerned with getting people out”, and too confused to provide relevant information about the building. The solution is to send firefighters to search for the required information. Therefore, they requested building information such as location of the pipe sprinkler systems (in Spanish, red seca and red inerte), structural building information (such as where the parking lots are), and evacuation routes. This information could be pre-loaded onto the application or shared through photographs.

- **Entrance routes.** Firefighters need information about entrance routes, e.g., to know the best street through which to get to the emergency site and “where truck Q2 will be, to know where we should park”.

- **Information about surroundings.** Contextual information is also important, for example fire hydrants, surrounding streets, subway stations, and photographs with relevant information.

- **Information about requests.** Frequently, firefighters at an emergency request the command center to send ambulances, policemen, and other services such as electric company workers. The command center requests the needed support but does not provide information about the status of the requests, so firefighters are sometimes unsure whether the support is on its way, already at the emergency site, or unavailable. This lack of feedback also causes other firefighter companies to request services that have already been asked for. One firefighter said, “this has happened frequently during calls... firefighters in charge request an ambulance but it already is on its way, and they ask for it again. We could get the information that it was already requested.”
we would free up the radio. Instead of asking about it through the radio, it would be better to just know”.

- **Traffic information.** Another suggestion was to add information about traffic, such as photographs or video streams of important streets, and information about accidents, closed streets, etc.

### Information sharing

Firefighters also commented on the PDA as a medium for communication, and how to use it to acquire the information they need. Some of their comments on this topic were the following:

- **Redundancy.** MobileMap is seen by the firefighters as a redundant medium to share information. One firefighter said, “We should remember the communication between the truck and the command center will not stop completely. We cannot suppose the firefighter will stop calling the command center altogether”.

- **Requesting information.** Some firefighters requested information to be sent from the command center, without having to download it. This prompted firefighters to discuss whether this would cause information overload, as the command center does not necessarily know the information that is most useful for a particular emergency. One firefighter commented “If fire is coming out of the eighth floor, the last thing on anyone’s mind is unneeded information. I think the best option is to contact the command center through the radio, say *I need this specific information*, and for the command center to send it.”

### Devices

We observed that firefighters are generally open and feel positive towards adding a PDA as a new communication device, but they also made it clear that they are busy with other tasks and cannot spend much time on operating a new device.

- **Managing another device.** The firefighter in charge of the truck rides in the front, next to the driver, and manages the information that he receives from the command center. However, he has to do several things at the same
time. One participant noted that it’s “very complicated for the volunteer in charge to be changing into his uniform, using the siren, with a PDA on his hand and talking on the radio.”

- **Devices on trucks.** The trucks in the two companies have laptop computers on board. One truck has a laptop in the back, connected to a screen in front. Another truck has a laptop computer in front, but “when we are on the road, and the laptop is open, the screen moves... we also pass through bumps on the way”. Most firefighters agreed that a good compromise would be to connect a PDA to a screen and generally use it to receive information while on the road.

**Usefulness**

Finally, we asked firefighters to discuss and evaluate how useful they found this proposal. Some of the comments were the following:

- **Emergency type.** The usefulness and usability of a device such as a PDA depends on the emergency type. One firefighter commented he would use it, “maybe not for a car crash, but for a fire”.

- **Time.** Most firefighters commented that MobileMap would reduce response and action time, if they could arrive at the scene already knowing some information, such as fire hydrants.

### 7.5.3.2. Second Focus Group

We now detail the information gathered from the second focus group, in which all participants had leadership positions, and therefore a different perspective than the first group. Results were also positive. Some comments overlapped with the first group’s, so we summarize here only the new information from this group.

**Information sharing**

- **Sharing photographs.** One of the participants belonged to a company specialized in hazardous materials (HAZMAT). He commented, “For my com-
pany, taking pictures would be great. For example, taking pictures of safety signs or the wind direction, the rest of the firefighters would be prepared”.

- **Sharing video.** Video could also be shared, by mounting a camera on top of a firetruck. In this way, firefighters commented they would be able to see where the truck is, its surroundings, and whether there are dangers around it.

Roles

Firefighters discussed who should be able to access information, and they commented on the differences between commanders and regular firefighters in regards to information access.

- **Filtering information for different roles.** Some commanders were concerned with some of the information being publicly available. They suggested some information is relevant for the person who is in charge of the fire truck, but should not be visible to the rest of the firefighters.

- **Identifying users.** The participants commented that each device should have a profile that could be loaded at startup to specify the user or the truck containing it.

### 7.5.3.3. Third Focus Group

We now detail the information gathered from the third focus group. Participants in this focus group were generally very enthusiastic, and had more questions than criticism. We summarize here only the new information from this group.

Roles

- **Firefighters not on duty.** A participant in this group brought up the problem of firefighters who are not on duty but may be available to work. Sometimes, a firefighter will ride on a fire truck practically on his own, so knowing whether other firefighters will show up at the emergency would be valuable. He commented, “Sometimes I’m in charge of the truck and I see only inexperienced firefighters are going. Then I listen to the radio and hear that the
emergency is a catastrophe, so it would be great to know if someone else is going, if five others are going, and it would be fantastic to know out of those five, who exactly is going”. Since the application would reduce radio communications, focus group participants proposed a web or cellphone-based interface to allow firefighters to access information about the emergency.

- **Communications administrator.** Participants were interested in adding information about the status of the emergency and any other relevant information. They suggested a new role could be created. This firefighter would be in charge of communications during the emergency, inputting information such as participants, a log of activities, etc. This information could be useful in case the commander changes, so when he takes charge he can read about the current situation instead of having to ask others or walk around the emergency.

### Security

- **Lost devices.** One participant was concerned with the data available through the application in case the device was ever lost. He suggested security measures such as “blocking information as it is being transmitted or completely blocking the system”.

- **Emergency details.** Currently, the command center may have some information about an emergency and choose not to share it through the radio because there it is not a private medium. Having secure communication between the command center and device could allow sharing more details about the emergency, e.g., the number and conditions of the people involved in a car accident, or the size of an explosion.

#### 7.5.4. Discussion

The results of the MobileMap evaluation in its three iterations are summarized in Table 7.21. The first row presents the number and percentage of requirements that were accomplished in each of the versions, out of the total number of requirements. We may note also that since MobileMap in its first two versions had only...
one identified user, the total number of requirements was only 14, whereas in the final two evaluations, the total requirements were more than 50. The second row includes the number of false positives, i.e., the number of requirements that the evaluation method proposed, and the number that were determined by the evaluator to actually be required. The interesting result here is in versions 3.0 and 3.1, in which 35.4% and 34.3% of the requirements were false positives. When we examined the evaluation results in detail, we found that out of these requirements, 64.7% and 63.6% of them were in the requirements between the server and other users. This suggests something that is also intuitive: requirements between a server and a user are not the same as between two users, and are actually much simpler. The third row displays the number of false negatives, i.e., the number of requirements that were determined to be requirements but were not suggested by the evaluation method. This number is 0 in all cases, which is a good result. We can see that the evolution of the software matches the evaluation results. The evaluation detects problems in collaborative support - for example, in version 3.0, the file exchange was not working, so the evaluation score was severely affected.

Cuadro 7.21: MobileMap evaluation statistics

<table>
<thead>
<tr>
<th>MobileMap</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>% accomplished</td>
<td>3/14 (21.4%)</td>
<td>2/14 (14.3%)</td>
<td>31/53 (58.5%)</td>
<td>37/51 (72.5%)</td>
</tr>
<tr>
<td>False positives</td>
<td>0/16</td>
<td>0/16</td>
<td>34/96 (35.4%)</td>
<td>33/96 (34.3%)</td>
</tr>
<tr>
<td>False negatives</td>
<td>0/16</td>
<td>0/16</td>
<td>0/96</td>
<td>0/96</td>
</tr>
</tbody>
</table>

Table 7.22 summarizes the main requirements proposed by the focus group, and how these requirements are also present in MCE evaluation. Several of the requirements can be aligned with issues proposed by MCE; however, there are also MCE suggestions that were not mentioned in the focus group, e.g. offline awareness.

Table 7.22 displays the general topics most discussed by firefighters, their suggestions, and how these suggestions are or are not present in MCE. For one part,
Cuadro 7.22: Summary of focus group results

<table>
<thead>
<tr>
<th>N</th>
<th>Topic</th>
<th>Suggestion</th>
<th>Relation to MCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Information availability</td>
<td>Building information</td>
<td>Not collaborative, therefore not possible suggestions through MCE evaluation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Entrance routes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Information about surroundings</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Information about requests</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Traffic information</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Information sharing</td>
<td>Redundancy</td>
<td>Flexibility</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Requesting information</td>
<td>Explicit data replication</td>
</tr>
<tr>
<td>3</td>
<td>Devices</td>
<td>Managing another device</td>
<td>Heterogeneity and interoperability, also related to usability issues not covered by MCE evaluation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Devices on trucks</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sharing photographs</td>
<td>File transfer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sharing videos</td>
<td>File transfer</td>
</tr>
<tr>
<td>4</td>
<td>Roles</td>
<td>Filtering information for roles</td>
<td>As stated by MCE evaluation, role identification is important.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Identifying users</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Firefighters not on duty</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Communications administrator</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Security</td>
<td>Lost devices</td>
<td>Security</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Emergency details</td>
<td>Ad-hoc work sessions</td>
</tr>
</tbody>
</table>
the focus groups validated the design of the last version of MobileMap: firefighters asked to identify separate roles, which have access to different functionalities (Topic 4). This means that the first versions of MobileMap, which only identified one role, are not adequate. Another result of the focus group was a great deal of requirements related to information availability and display, and therefore not related to collaborative support (Topic 1). They also identified the importance of heterogeneity and interoperability (Topic 3) through discussing the devices they would operate. These requirements, though useful to improve the software, do not affect the results of MCE since they do not deal with collaboration.

Regarding collaborative support, firefighters also commented on several implemented features, such as automatic connectivity, since they have little time to dedicate to configuring the application. They also found synchronous messaging and file transfer to be useful (Topic 3), and several commented on the importance of keeping data secure (Topic 5). We can therefore observe that several requirements that were identified by MCE were also identified by the potential users during focus groups, with no explicit prompting (users were not explicitly asked to discuss these requirements).

The focus group also confirmed that it is difficult for users to express fine-grained collaborative issues, which confirms the motivation behind this thesis. This is shown by the fact that users did not comment on several issues that have been identified in the literature as central to collaboration, e.g. awareness [38]. The MCE method is therefore valuable to uncover these hidden requirements.

Finally, we also can observe that the needs of the firefighters are aligned with the third version of the software, which confirms that the evolution of the software has been positive.
7.6. MCE Method Cost

Section 5.3 presented the cost of the MCE method in a graph that compared it to the cost of other evaluation methods. This section presents empirical data to further study hypothesis H2, which is concerned with the cost of the proposed evaluation method in regards to existing methods.

Cuadro 7.23: Cost of applying MCE to studied scenarios

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Effort Activities</th>
<th>Effort People</th>
<th>Time G.1- G.3</th>
<th>Time G.4- G.5</th>
<th>Time E.1- E.4</th>
<th>~Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Urban Search and Rescue</td>
<td>Interviews, observation, modeling</td>
<td>Users, evaluator</td>
<td>~10 hours</td>
<td>1 day</td>
<td>N/A</td>
<td>Days</td>
</tr>
<tr>
<td>Construction Inspections</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>6 hours</td>
<td>Hours</td>
</tr>
<tr>
<td>Healthcare</td>
<td>Shadowing study, modeling</td>
<td>Users, evaluators</td>
<td>2 days/2 weeks</td>
<td>1 day</td>
<td>1 day</td>
<td>Days-Weeks</td>
</tr>
<tr>
<td>Daily emergency management</td>
<td>Interviews, modeling</td>
<td>Firefighter captains, evaluator</td>
<td>6 hours</td>
<td>1 day</td>
<td>1 day</td>
<td>Days</td>
</tr>
</tbody>
</table>

The cost of MCE for each of the studied scenarios is summarized in Table 7.23. Cost is decomposed into effort and invested time. The effort category details the activities that were undertaken to conduct evaluation and the people involved in them. The time category details the time taken to carry out steps G.1 to G.3 (analyze the collaborative situation), G.4 to G.5 (build the graph with the collected information) and E.1 to E.4 (generate requirements and evaluate their presence in the system). Some information is not available (N/A), e.g. steps E.1 to E.4 were not carried out in the USAR scenario, and some steps were done by the application developer in case of construction inspections. The final column adds up the total time, to check its corresponding category (either hours, days or weeks).

This study of the costs leads us to the following observations:
• **Time is variable.** Depending on the particular scenario and activities chosen to study it, time for steps G.1 to G.3 can vary from a few hours to a few weeks. In the case of the healthcare scenario, observations amounting to two days were recorded over the course of two weeks, so the longer time period was considered, even though the effort was not constant and sustained during the whole time period. Naturally, the more time spent conducting interviews, focus groups and observations will lead to a more thorough understanding of the collaborative work, but it will also increase the cost. On average, to conduct MCE, a few days seems to be enough time to understand the collaborative work scenario.

• **Users are involved.** Even though in the theoretical analysis users were not considered to be involved, we can see from the data that users were involved in interviews and working while being observed. This may increase the cost in case users have to be interrupted from their work, which was the case in e.g. the healthcare scenario. Depending on the setting, evaluators may choose the least disruptive option to analyze collaborative work, in order to keep costs low. In the cases discussed here, analysis was conducted with goals other than evaluation, so the chosen activities were not necessarily low cost (e.g. the shadowing study).

• **Evaluation and analysis.** Generally, evaluation should not be considered a standalone activity, rather, it should be integrated into the development process. This means that analyzing the scenario will not usually be done solely with the goal of evaluation, so the costs of this stage may be shared with the other activities of the development cycle.

The data validates the theoretical analysis of the costs of MCE. However, there is more variability than what was foreseen in the previous analysis. Even so, this variability can be managed to prevent costs from increasing too much, and data from the analysis phase of development can be reused for evaluation, lowering its costs. This concludes the proof of the second hypothesis, since we have shown that the presented evaluation method has an equal or lower cost to existing methods.
7.7. Discussion

The experiments detailed in this chapter show that the MCE method is usable and useful to analyze and evaluate mobile collaborative applications in loosely coupled work settings. Naturally, as we conduct more evaluation processes, we will understand even more when MCE is the best choice. However, the obtained results are encouraging, and show a correlation between perceived collaboration support and evaluation results.

The MCE method does have some limitations. For example, it is focused solely on loosely coupled mobile group work. This leaves out tightly coupled collaboration scenarios such as collaborative note-taking in classrooms. There is also a tradeoff with regards to the fact MCE does not require users. We confirmed several difficulties in user-based evaluation in our experiments with MobileMap and in brainstorming systems evaluation, e.g. finding a heterogeneous group of users, getting users to be critical and think about their work practices, etc. In this sense, conducting evaluation in a laboratory setting and by studying how users work is simpler, easier, and can even be done to propose new collaborative work settings. Of course, the lack of user opinions could impact the usability and acceptance of the final product. If we compare MCE to existing evaluation methods, it is probably close to groupware heuristic evaluation (GHE), or to the heuristic evaluation method for conventional systems [105]. GHE is focused on real-time, shared workspace groupware systems [12], and MCE is aimed at mobile shared workspace systems. However, they share some drawbacks, such as the fact “surprise”, or unexpected, problems may not be found because users are not involved in the evaluation. This problem is solved through our previous discussions on how to organize an evaluation process, which concluded that evaluators must understand the focus of their evaluation and combine evaluation methods to get a comprehensive result.

Additional data would help answer several interesting questions. First, we can assume that during the implementation of a software, developers prioritize important functional requirements, leaving less urgent requirements for subsequent development. If we had more data from the first efforts at developing software,
we could gather information about the most important requirements, and use this information to suggest a priority level for each requirement. This could be helpful for the first phases of new software development.
Capítulo 8

Conclusions and Future Work

This chapter concludes the thesis, presenting the final thoughts on this work. First, a summary of the main topics of this dissertation is presented. Second, we highlight the contributions of this work to the area of research in mobile collaboration. Then, we present the lessons learned, some concluding remarks and the directions for future research.

8.1. Summary of research

This thesis centers around the problem of collaborative software evaluation. There are several difficulties related to evaluation in this type of systems. Some of the main difficulties are the complexity of work processes, the high expense of involving users, and the existence of several external factors influencing collaboration, such as personalities, group cohesion and organizational context.

This work introduced an evaluation method tailored specifically to mobile collaborative systems in loosely coupled scenarios, and an evaluation methodology to organize the evaluation process of a collaborative system. The evaluation method has two dimensions: a modeling language to analyze mobile collaborative scenarios, and a list of requirements based on existing literature. The characterization of loosely coupled mobile work and the modeling language were based on my observations of the work done at one USAR simulation. Each graph conveys information
8.1 Summary of research

about the involved roles and the corresponding scenarios when they need to in-
teract. This can help design a system, i.e., the evaluation method can suggest
requirements depending on the scenario. For example, two users who need to co-
llaborate while in an unreachable situation can be helped through mechanisms to
notify them when they become reachable again.

The graphs have several interesting properties. First, they can be validated to
check if they are coherent, which can help a designer make sure he understands
the scenario he is modeling. Second, the graphs can evolve as understanding of a
scenario improves or as requirements change. Also, any information that can be
associated to the interaction scenario of two users can be automatically generated
from the graph for the scenario.

MCE is a formative evaluation method, which means it can be used at any
point of the development process of a mobile shared workspace to support loosely
coupled work. To conduct evaluation, only an analysis of the collaborative process
(obtained e.g. through interviews, focus groups, or observations), and the software
at any point in development (conception, analysis, implementation) are necessary.
During the course of this research, an unexpected use as a requirements tool was
discovered, even though MCE is focused solely on evaluation.

The modeling language and evaluation method were tested in several settings.
First, communication in USAR operations was studied through audio recordings
of a complete simulation. The results of this study were used to further examine
the proposed characterization of collaborative work. Second, two independent sce-
narios, construction inspections and healthcare, were used to apply the evaluation
to new areas and study whether the evaluation method could help measure and
predict collaborative system support in them. Finally, MCE was applied to three
iterations in the development of MobileMap, an application to support everyday
emergencies. These results were contrasted with evaluation results from several fo-
cus groups. The test results were encouraging. Naturally, further research can help
determine other work scenarios in which the evaluation method can also be applied.
8.2. Contributions

This research focused on the evaluation process of mobile shared workspaces in order to improve the collaborative system functionalities and fulfill the users’ expectations. The main contribution of this thesis is an evaluation method specifically designed for mobile collaborative systems in loosely coupled work settings. The main and secondary contributions of this thesis are detailed in this section; they are the following ones:

- Evaluation method
- Methodology for organizing the evaluation process
- Technique for evaluation cost comparison
- Software tool to support the evaluation process
- Understanding of urban search and rescue scenarios

8.2.1. Evaluation method

The main contribution of this thesis is an evaluation method designed specifically for mobile collaborative systems. This is an area that is largely unexplored, although there is consensus that mobile and collaborative systems evaluation is difficult. The evaluation method is based on a characterization for loosely coupled mobile collaboration, a modeling language and a list of requirements. All of these aspects are part of the main contribution of this thesis.

8.2.2. Methodology for organizing the evaluation process

This dissertation proposes a nomenclature for distinguishing types of evaluations between tools, methods, and frameworks. Although the main contribution is an evaluation method, an evaluation framework was proposed to organize an evaluation process. There are actually very few concrete guidelines for this subject in collaborative systems.
8.2.3. **Technique for evaluation cost comparison**

Previously, some evaluation methods were said to be “low cost”, or “discount”, but there was no way of really comparing the cost of collaborative system evaluation methods. One of the goals of this thesis was to define a metric to provide a way to compare them, but it was found that an exact metric is not applicable because many factors contribute to the total cost of an evaluation method, and the weight of each may be different for each case. Therefore, we defined the two dimensions that contribute to cost: time and effort, and while studying existing methods, found the factors that were said to be most influential in their cost (e.g. including users, modeling). The technique for evaluation cost comparison is based on graphically displaying methods according to their time and effort, based on the previously mentioned factors. This technique is purposefully not exact, so it allows for the variations in weight for each factor. In any case, it is successful in organizing and comparing evaluation method costs, providing a clear visualization and comparison that can help select an appropriate evaluation method for a given situation.

Evaluation methods that are meant to be used in a formative way should be low cost; otherwise, the high amount of effort and time involved would hinder future evaluations. Naturally, low-cost methods can be mixed with higher cost methods that are applied sparingly to produce a comprehensive evaluation.

8.2.4. **Software tool to support the evaluation process**

A tool for helping evaluate mobile shared workspaces was implemented to help ease the task of conducting evaluation. This software tool allows the evaluator to model the work scenario, and automatically validate it and generate its requirements. Then, the evaluator can mark those requirements that have been successfully implemented in the software and track their progress.
8.2.5. Understanding of urban search and rescue scenarios

This thesis also contributes to provide a clearer understanding of a type of mobile collaborative work, which is urban search and rescue settings. This setting had been previously studied by some authors, but our access to simulations and to the radio communications improved our understanding of this type of work and may help researchers design systems for this scenario in the future.

8.3. Lessons Learned

This section presents reflections on a few of the lessons learned during the development of the thesis. These lessons will be used when undertaking any new research project. They are the following ones:

- **Importance of data**: The development of this thesis highlighted the importance of gathering relevant data on which to base analysis. For instance, the first observation of firefighters working in USAR operations was conducted with a notepad and simple observations and photographs, while the second one consisted of the same, plus video and audio recordings. Irrelevant information from these recordings was filtered later on, but the raw data provided valuable information.

- **Clear, unambiguous definitions**: Perhaps one of the most difficult points in the development of this dissertation was finding appropriate definitions for reachability and simultaneity. The first definition used for reachability caused confusion when it was shared with developers and applied to the scenarios they were working in, and their help was extremely useful to clarify the definition and make it unambiguous.

- **Tradeoffs**: When designing a modeling language, dealing with tradeoffs is difficult. The simplicity of the language is important, and so is an opposing force: modeling the situation as faithfully as possible. The lesson learned was to try to find the balance between both forces, through a process of trial and error.
8.4. Conclusion

Mobile devices have become increasingly widespread, and MSW can incorporate them into diverse areas that require mobile collaboration. This thesis provides an increased understanding of mobile collaboration and tools for analysis and evaluation of systems to support it. These contributions may positively impact the software industry by easing the development of mobile shared workspaces, which should in turn promote the creation of this type of system. The future arrival of mobile collaborative software fulfilling organizational needs and developed in an easier, less expensive and more effective way than current systems may encourage their adoption by groups whose work could be improved. Workers in mobile collaborative scenarios in diverse fields such as healthcare, education and tourism, may benefit from systems to support their work processes, since these systems can improve e.g. communication, and awareness of others’ work. The involved organizations, in turn, may also benefit from the effects of mobile collaborative software such as improved efficiency. We hope to contribute a new way to look at pervasive collaboration scenarios and understand them, with the goal of building systems that truly support collaborative processes.

8.5. Future work

This dissertation has opened up several paths to new research in the area of collaborative mobile systems evaluation. This section details the topics for future work, which are the following ones:

- Refine requirements for intermediaries
- Usability of modeling language and software tool
- Framework for mobile shared workspaces

8.5.1. Refine requirements for intermediaries

Servers or passive intermediaries in collaboration were considered simply as another collaborator in this work. However, the results we obtained suggest ser-
vers in this type of scenario have characteristics and requirements particular to them. Future work should elaborate on the type of requirements and services a server should provide to users.

8.5.2. Usability of modeling language and software tool

The modeling language, used through the Graph Modeling Tool (GMT), was evaluated only superficially. Three independent users worked with GMT and provided informal feedback, which helped improve the software and correct deficiencies. However, this is only an initial test of whether the modeling language is easy to use, and we also need to study other usability aspects such as how long it takes for a developer to become proficient in the language and able to benefit from it.

8.5.3. Framework for mobile shared workspaces

It is natural to observe that the presented evaluation method resembles a mid-level framework for building mobile applications. This is an interesting byproduct of this research: we could provide modules to implement requirements, to take advantage of the similarities found in mobile work settings.
Apéndice A

Further information

This chapter presents further details about two case studies presented in the experimental results of the thesis: urban search and rescue and construction inspections. The first section details further results from the observation and recording of firefighters at work. The second section presents the detailed results of applying MCE to the construction inspections application.

A.1. Urban Search and Rescue

The first simulation was approximately two hours long. The exercise was observed and notes were taken to get a general idea of the collaborative process and the sequence of events. During the exercise, firefighters split work into three general tasks, which were: finding victims in a collapsed structure (rubble), lifting heavy objects to rescue victims or clear the area, and constructing support for damaged structures. Figure A.1 shows the firefighters’ training grounds, where the simulation of the emergency took place. On the right hand side, a diagram displays the layout of the emergency site and how resources and units are organized outside of it. The observations of this first exercise are detailed in Appendix A.

Some people stayed outside of the emergency site to organize resources and plan strategies to deal with the emergency. They were the incident commander, medics, logistics, communications, and the security officer (or entry control). Other
The events of the simulation were the following.

1. The group of firefighters is split up into several sub-groups and assigned specific tasks. Each group has an assigned team leader. Radios are distributed to each of the groups.

2. Groups leave to work on their assigned tasks. A person in charge of entry control checks the equipment and identity of each firefighter entering the disaster area.

3. If an earthquake replica or other emergency occurs, an alarm sounds and everyone gathers in a designated safety area to make sure each firefighter is accounted for. This information is transmitted by radio.

4. All information is sent to the incident commander. Some information is urgent (a firefighter is missing after an earthquake replica) and other information is not (the medic calls the IC on the radio to let him know that one of the rescued victims is being transported to the hospital). Several times, the IC tries to
communicate with one of the other firefighters and is not heard. Sometimes, the firefighters must repeat a message that is not well understood due to problems with the transmission.

5. The IC tracks the progress of the firefighters (how many victims have been rescued, how many are left to be rescued, etc.). The operations end when all the victims have been rescued and all the firefighters are safely outside the emergency site.

A.1.1. Description of work

There are three phases to a search and rescue operation: size up, search, and rescue. This information was obtained from interviews and from a USAR training course manual [18].

Size up

The first phase is a size up, in which firefighters gather facts and make decisions on the course of action. Operations sends out reconnaissance personnel into the area, who must identify individual structures, perform general hazard assessment, and search for trapped victims. There are three steps to this process:

1. Identification and description of buildings for reference
2. Rapid assessment of the affected area
3. Identification of locations that require a more detailed assessment

The emergency area must be characterized, to have a localization reference for the site. Firefighters draw the street of the emergency site, identifying the street name and each building with a number (Fig. A.2(a), A.2(b)). Each building is then characterized: floors are numbered, and each side is labeled (Fig. A.2(c), A.2(d)).

In this step, a preliminary organization of the situation takes place, in which firefighters determine which resources are needed and where, and may organize medical help, etc. The incident commander begins to understand the complete situation with the gathered information.
Figura A.2: (a) Known building numbers are used (e.g., 701 and 706), unknown ones are filled in. (b) If all numbers are unknown, numbers are assigned for reference. (c) The sides of buildings are numbered. (d) A building is divided into four quadrants.
A.1 Urban Search and Rescue

Search

Firefighters then make a plan to determine the search strategy, which should focus on locating the greatest number of victims in the shortest amount of time. Then, firefighters engage in search operations, which consist of interviewing survivors and a systematic movement across the site while listening to calls for help, and may be aided by canine search and electronic search. Marking systems may be used to label buildings with an evaluation of the situation inside; the marks indicate which buildings have already been searched, the results of the search; they help avoid duplication of search efforts. An example of the marking system for structures is shown in Fig. A.3.

Marks are also used to inform others about the location of victims who are not immediately rescued. This is marked with a V near the victims’ location. L indicates the number of live victims; D indicates the number of dead victims. An arrow indicates that the location of the victims has been confirmed. If only dead victims remain, a horizontal line will be drawn crossing the V. When all victims have been rescued, a circle is drawn around the V.

Rescue

Rescue follows search operations and is focused on extricating the greatest number of victims in the shortest amount of time. The team must determine which resources to commit to a rescue site based on the potential success, prioritizing...
rescues that are easily achievable and then moving on to more complex ones. In places with a high probability of survivors, rubble is selectively removed to search for victims. When there is certainty there are no more live victims, authorization is granted to move large quantities of rubble to find the remaining victims.

A.1.2. Communication in USAR

Communication during the emergency in general takes three forms. Inside the teams, firefighters communicate verbally to synchronously solve the problems they are dealing with. Each team works autonomously from the other groups, but when they need to communicate (report a victim has been found, ask for help, etc.) they use the radio. The radio is by far the most frequent form of communication among team leaders, the operations officer, the incident commander and other firefighters outside the emergency area. The third communication medium is an acoustic signal or alert, which is used for urgent communication. One long blast (3 seconds) is meant to convey “All quiet, search will begin on a predetermined area”. Two short blasts are used for “Search over, resume operations” and are only used by the group leader. Three short blasts is the signal for “Danger, evacuate the area”, which can be used by any member of a team who sees an abnormal situation.

Message frequency was studied. Fig. A.4 groups radio messages every four minutes, and it displays the number of messages sent, the message length (number of words) of each message, and message length in radio time. We can see that the radio is constantly used throughout the emergency, only decreasing at the end when the emergency was over and the firefighters were organizing the exit of the emergency area (a small peak near the end was one of the firefighters becoming injured and having to be rescued). The average radio use is 150 seconds every 4 minutes, i.e., the radio was busy at least 62.5% of the time (this is a conservative estimate that does not count interferences, unintelligible messages, and times the radio was not available due to audio alarms). The audio from the first few minutes was not coded due to poor quality. It is important to observe that grouping data in intervals of other lengths results in a graph with the same general shape.
A.2 Construction Inspections

This section presents the results of applying MCE to the construction inspections application in Tables A.1 and A.2.

Figura A.4: Radio communication
### Cuadro A.1: Requirements for Construction Inspections - Table 1

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Foreman</th>
<th>Contractor</th>
<th>Inspector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chief inspector</td>
<td>Req?</td>
<td>Done?</td>
<td>Req?</td>
</tr>
<tr>
<td>Ad-hoc work sessions</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Asynchronous messaging</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Automatic connection</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Automatic peer detection</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Caching</td>
<td>Yes</td>
<td>Undecided</td>
<td>Yes</td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Explicit data replication</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>File transfer</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Message routing</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Pushing notifications</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Offline awareness</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Online awareness</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Synchronous messaging</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Transition awareness</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User connection/disconnection</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User gossip</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### Cuadro A.2: Requirements for Construction Inspections - Table 2

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Contractor</th>
<th>Inspector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ad-hoc work sessions</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Asynchronous messaging</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Automatic connection</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Automatic peer detection</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Caching</td>
<td>Yes</td>
<td>Undecided</td>
</tr>
<tr>
<td>Conflict resolution</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Explicit data replication</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>File transfer</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Message routing</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Pushing notifications</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Offline awareness</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Online awareness</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Synchronous messaging</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Transition awareness</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User connection/disconnection</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>User gossip</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
References


REFERENCES


170
REFERENCES


171
REFERENCES


175
REFERENCES


REFERENCES


177


178
REFERENCES


179


