



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

**ENGINEERING INTERACTION REQUIREMENTS OF MOBILE APPLICATIONS THAT
SUPPORT PEOPLE-DRIVEN COLLABORATIVE PROCESSES**

TESIS PARA OPTAR AL GRADO DE DOCTOR EN COMPUTACIÓN

MAXIMILIANO CANCHÉ EUÁN

PROFESORES GUÍA:

SERGIO OCHOA DE LORENZI
DANIEL PEROVICH GEROSA

MIEMBROS DE LA COMISIÓN:

JOSÉ PINO URTUBIA
ALEXANDRE BERGEL
AURORA VIZCAÍNO BARCELÓ

Esta tesis fue financiada por PRODEP (México) - PROMEP/103.5/16/6096, parcialmente por CONICYT-PCHA/Doctorado Nacional/2019 - 21191825, y parcialmente por Fondecyt (Chile), 1191516.

SANTIAGO DE CHILE

2023

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE DOCTOR EN COMPUTACIÓN
POR: MAXIMILIANO CANCHÉ EUÁN
FECHA: 2023
PROFESORES GUÍA: SERGIO OCHOA,
DANIEL PEROVICH

INGENIERÍA DE REQUERIMIENTOS DE INTERACCIÓN PARA APLICACIONES MÓVILES QUE APOYAN PROCESOS COLABORATIVOS DIRIGIDOS POR PERSONAS

Los participantes de un *proceso colaborativo dirigido por personas* (PDCP en inglés) usualmente requieren una aplicación colaborativa móvil de apoyo a sus actividades. Dado que el flujo de trabajo de un PDCP no está definido al momento de su diseño, concebir estos sistemas no es sencillo debido a las diversas necesidades de interacción de los participantes, y al escaso conocimiento para ayudar a los diseñadores a abordar los servicios de coordinación y colaboración en etapas tempranas del desarrollo. Esto significa que los desarrolladores y stakeholders tienen que lidiar con productos ad-hoc en proyectos a la medida. En el diseño de estos productos, los desarrolladores especifican principalmente los escenarios de interacción a ser apoyados con la aplicación móvil, así como los servicios de interacción que utilizarán los participantes. Tales especificaciones (generalmente modelos visuales) deben ser analizadas, validadas y refinadas por los desarrolladores y stakeholders, con el fin de llegar a un acuerdo temprano sobre el alcance de la aplicación móvil en desarrollo.

Un estudio realizado en este trabajo muestra que los lenguajes y notaciones usados para crear modelos visuales de los escenarios de interacción, están limitados para abordar los desafíos involucrados en esta actividad de modelado. En consecuencia, esta tesis propone un lenguaje denominado CIMoL (*Computer-Mediated Interactions Modeling Language*) para modelar escenarios de interacción mediados por computadora involucrados en PDCPs. Este lenguaje, junto con una herramienta de generación automática de prototipos (denominada *CIMoL Modeler*), permite a los ingenieros de software y stakeholders identificar y validar los servicios de interacción y awareness que requieren los participantes en un PDCP.

La evaluación de CIMoL involucró: un estudio de expertos considerando diversos aspectos de diseño y un estudio de caso donde los participantes concibieron sistemas colaborativos móviles para apoyar un PDCP particular. Los resultados fueron positivos, lo que indica que CIMoL puede ayudar efectivamente a los desarrolladores y stakeholders a co-concebir, validar y refinar los escenarios de interacción, y a lograr un entendimiento compartido sobre el alcance del sistema en desarrollo. Como trabajo futuro, se propone una evaluación adicional de CIMoL, que involucre a un grupo más grande de participantes y PDCP en dominios comerciales adicionales. Así identificar oportunidades de mejora y su nivel de madurez para la adopción en la industria.

ABSTRACT OF THE THESIS TO OPT FOR
THE DEGREE OF DOCTOR IN COMPUTING
BY: MAXIMILIANO CANCHÉ EUÁN
DATE: 2023
ADVISORS: SERGIO OCHOA, DANIEL
PEROVICH

ENGINEERING INTERACTION REQUIREMENTS OF MOBILE APPLICATIONS THAT SUPPORT PEOPLE-DRIVEN COLLABORATIVE PROCESSES

Participants involved in a *people-driven collaborative process* (PDCP) usually require a mobile collaborative application to support their activities. Given that the workflow of a PDCP cannot be established at design time, conceiving these mobile systems is far from simple, in part due to the diverse interaction needs of the participants and the scarce knowledge to help designers address the coordination and collaboration services at early stages of the system development. It means that developers and stakeholders have to deal with ad hoc products in bespoke projects. In the design of such products, the developers mainly specify both the interaction scenarios to be supported with the mobile application and the interaction services to be used by the participants to carry out the process. Such specifications (typically visual models) should be analyzed, validated, and refined by developers and stakeholders in order to reach an early agreement about the scope of the mobile application being developed.

A study conducted in this thesis work shows that existing languages and notations, used to create visual models of the interaction scenarios, are limited to address the challenges involved in this modeling activity. Therefore, this dissertation proposes a language named *CIMoL (Computer-Mediated Interactions Modeling Language)* for modeling computer-mediated interaction scenarios involved in PDCPs. This language, together with an automatic prototyping tool (named *CIMoL Modeler*), allows software engineers and stakeholders to identify and validate the interaction and awareness services required by the participants in a PDCP.

Two instruments were used to evaluate CIMoL: an expert study considering a set of design aspects reported in the literature and a case study where the participants conceived mobile collaborative systems to support a particular PDCP. The results were positive, indicating that CIMoL effectively can help developers and stakeholders to co-conceive, validate and refine the interaction scenarios, and then reach a shared understanding about the scope of the system being developed. As future lines of research, we propose a further evaluation of CIMoL involving a larger group of participants and PDCPs in additional business domains, to identify both opportunities for improvement and the level of maturity for adoption in the industry.

to my family

Acknowledgements

This thesis wouldn't be possible without the support and guidance from a lot of people. It is my pleasure to mention some of them here. I would like to express my gratitude to Sergio Ochoa and Daniel Perovich for the excellent mentoring/supervision throughout the entire PhD thesis.

My advisory committee played a significant role in shaping and revising the final thesis, and I am grateful for their time and advice. The committee members were José Pino, Alexandre Bergel, and Aurora Vizcaíno.

I owe a special thanks to my professors and friends from the University of Chile, especially Rolando and Fran. Without them, last months wouldn't have been so fun and memorable.

I would like to thank the researchers from the University of Granada, Spain and the University of Cauca, Colombia for their interest, ideas, and criticism.

But there are a lot of other people I would like to thank for their support: María Cecilia Bastarrica, Jocelyn Simmonds, Gonzalo Navarro, Hernán Cuevas, Angélica Aguirre, Sandra Gáez, Luz Ceballos, Ren Cerro, José Luis Garrido, Valeria Herskovic, Rodrigo Santos, Marco Gerosa, Juliana França, Mariem Ben Hassen, Zaharah Bukhsh, Christoph Mayr-Dorn, Birger Lantow, among others.

Table of content

Chapter 1. Introduction	1
1.1. Motivation	1
1.2. Conceiving Mobile Collaborative Applications to Support PDCPs	3
1.3. Problem Definition	5
1.4. Work Hypotheses	7
1.5. Research Goals	7
1.6. Research Methodology	8
1.7. Document Structure	9
Chapter 2. Background	11
2.1. Collaborative Processes Structuring	12
2.2. Mobile Collaborative Systems that Support PDCPs	13
2.2.1. Using Mobile Collaborative Applications in PDCPs	14
2.2.2. Types of Participants in a PDCP	16
2.3. Strategies for Developing Mobile Collaborative Systems	17
2.3.1. Development Focused on the System Design	17
2.3.2. Development Focused on the Engineering of the System	20
2.3.3. Hybrid Development Processes	22
2.4. Product Scoping	24
2.4.1. Relevance of the Early Definition of the Product Scoping	24
2.4.2. Product Scoping in Bespoke Projects	26
2.5. Requirements Elicitation for Collaborative Systems	28
2.5.1. Perspectives of Elicitation Techniques for Collaborative Systems and their Contribution to Define the Product Scope	29
2.5.2. Requirements Elicitation for Collaborative Systems Supporting a PDCP	31
2.6. Shared Understanding between Stakeholders and Software Engineers	31
2.6.1. Types of Shared Understanding	32
2.6.2. Practices to Achieve a Shared Understanding	32
2.6.3. Relationship between Explicit Specifications and Explicit Shared Understanding	34
2.7. Designing Effective Visual Modeling Languages	34
2.7.1. The Importance of Visual Representations	35
2.7.2. An Approach for the Development of Modeling Languages	35
2.7.3. Design Principles for Cognitively Effective Visual Notations	37
2.7.4. The notion of complexity in visual modeling	39
2.8. Model Consistency	40
2.8.1. A Classification of Model Consistency	40
2.8.2. Types of Consistency Checking Techniques	40
2.8.3. Main Consistency Checking Techniques Used in Existing Studies	41

2.8.4. Tool Support for Consistency Checking	41
2.9. Summary	42
Chapter 3. Related Work	43
3.1. Modeling Languages Description	43
3.1.1. Case Management Model and Notation	43
3.1.2. BPMN Plus	45
3.1.3. BPMN4SBP	46
3.1.4. Little-JIL and hADL	47
3.1.5. Mobile Collaboration Modeling	49
3.1.6. IoT Modeling Notation	50
3.1.7. Collaboration Graphs	51
3.1.8. Computer-mediated Interaction Modeling Notation	52
3.2. Expert Review of Model Representations	53
3.3. Discussion	55
3.4. Threats to Validity	55
3.5. Summary	56
Chapter 4. The Computer-Mediated Interactions Modeling Language	57
4.1. Language Foundations	57
4.2. Language Metamodel	58
4.3. CIMoL Common Constructions	59
4.4. CIMoL Notation for the Developer View	61
4.5. Interaction Services Specification	63
4.6. Examples of Interaction Graphs	66
4.7. Summary	68
Chapter 5. Evaluation of the CIMoL Capabilities	69
5.1. Study Settings	69
5.2. Definition of the Assessment Instruments	70
5.2.1. Instrument for Assessing the Understandability of the CIMoL Visual Notation	70
5.2.2. Instrument for Assessing the CIMoL Capability to Represent Computer-mediated Interaction Scenarios	72
5.3. Assessment Preparation	74
5.3.1. Documents Preparation	74
5.3.2. Participants in the Assessment	74
5.4. Introduction to the Assessment Activity	75
5.5. Language Assessment Process	75
5.5.1. Dynamic of the Assessment Process	75
5.5.2. Exploring the Language	76
5.5.3. Collecting the Experts' Opinion	77
5.6. Analysis of Results	78

5.6.1. Concerning the Understandability of the Notation	78
5.6.2. Concerning the Suitability of CIMoL to Specify Computer-Mediated Interaction Scenarios	84
5.6.3. Comparing CIMoL with the Related Work	85
5.6.4. Threats to Validity	87
5.7. Summary	87
Chapter 6. CIMoL Modeling Tool	89
6.1. Requirements for the Modeling Tool	89
6.2. Structure of the CIMoL Modeler	90
6.3. Modeling the Process Structure	91
6.4. Modeling Interaction Scenarios	94
6.4.1. Modeling Roles and Interactions	96
6.5. Modeling Prototypes	98
6.5.1. Home Mockup	99
6.5.2. Notifications Mockup	100
6.5.3. Contacts Mockups	100
6.6. Models Consistency Checking	103
6.7. Automatic derivation of interaction services	104
6.8. Where the Tool Should Be Used	105
6.9. Who Should Use the Tool and How to Use It	105
6.10. How to Model an Interaction Graph Using the Modeling Tool	107
6.11. Summary	108
Chapter 7. Case Study: Using CIMoL to Model an Elderly Caregiving Supporting System	109
7.1. Case Study Design	110
7.1.1. Research Questions	110
7.1.2. Introduction to the Informal Elderly Caregiving Process	110
7.1.3. Selection of Participants	111
7.1.4. General Description of the Activity	112
7.1.5. Data Gathering Mechanisms	114
7.1.6. Data Analysis Approach	115
7.2. Case Study Execution	115
7.2.1. Replica I: Elderly with Senile Dementia and Mobility Limitations	115
7.2.2. Replica II: Elderly with Down's Syndrome	119
7.2.3. Replica III: Elderly with Mobility Limitations	122
7.2.4. Replica IV: Elderly with Mobility Limitations	124
7.3. Discussion	126
7.4. Threats to Validity	128
7.5. Summary	128

Chapter 8. Conclusions and Future Work	130
8.1. Goals Achievement	130
8.2. Contributions to Theory and Practice	131
8.3. Limitations	132
8.4. Future Work	133
Glossary	135
Bibliography	136
Annexes	156
Annex A. Instrument for Stakeholder - Replica I	156
Annex B. Instrument for Engineers - Replica I	157
Annex C. Instrument for Stakeholder - Replica II	158
Annex D. Instrument for Engineers - Replica II	159
Annex E. Instrument for Stakeholder - Replica III	160
Annex F. Instrument for Engineers - Replica III	161
Annex G. Instrument for Stakeholder - Replica IV	162
Annex H. Instrument for Engineers - Replica IV	163
Annex I. Analysis of Notations for Modeling User Interaction Scenarios in Ubiquitous Collaborative Systems	164
Annex J. List of Publications	178
Annex K. Technical Reports	180

Chapter 1. Introduction

1.1. Motivation

According to Dorn & Dustdar (2011), a *people-driven collaborative process* (PDCP) involves participants playing several roles, but with no pre-established workflow that coordinates the actions and interactions among roles. Therefore, the participants decide their next action on-the-fly considering the status of particular context variables, for instance, their to-do list, the priority of their pending activities, or the availability of their counterpart. Examples of people-driven collaborative processes are *agile software development* (Beck et al., 2001), *global software development* (Portillo-Rodríguez et al., 2012), *disaster relief efforts* (Aldunate et al., 2006), *hospital work* (Pinelle & Gutwin, 2006), *integrated business activities* (Hawryszkiewicz, 2009), *construction inspections* (Ochoa et al., 2011) and *elderly caregiving* (Cruz et al., 2016; Gutierrez & Ochoa, 2017).

The PDCPs usually involve individual and group goals. The former are related to the roles played by the participants, and the latter goals are linked to the outcomes these people have to reach as a team. Typically, the goals and work context of the participants establish the dynamic of the process, and also their interaction needs.

When the work in PDCPs is addressed using technology, each participant and his/her supporting application (usually a mobile collaborative system) becomes a *human-machine unit* (HMU). These units interact with others to perform the PDCP, trying to reach individual and group goals.

Figure 1.1 shows two interaction scenarios where the participants (i.e., the HMUs) perform several activities to carry out a PDCP. In the first scenario, the inspectors are deployed into a construction site to check the status of the facilities in a particular area. During these inspections, the participants (i.e., the inspector-in-chief, the foreman and the inspectors (Ochoa et al., 2011)) work individually, and collaborate on-demand when required; for instance, when a second opinion about the infrastructure status is required. Once the information gathering in the inspected area finishes, the participants usually meet to cross-check their individual diagnosis, solve discrepancies and communicate the results to each other, to the foreman and to the inspector-in-chief.

A similar dynamic is followed by nurses, physicians and paramedics when they take care of hospitalized patients (Fig. 1.1.b). These people are responsible for several aspects of the healthcare process (i.e., these HMUs play particular roles in the PDCP), therefore, they have individual goals, but also group goals. These latter goals require participants to perform coordinated work to make the healthcare process more effective.



Figure 1.1. Interaction scenario in a PDCP process: (a) construction inspection process, and (b) hospital work.

The capability and flexibility to perform interactions among HMUs make a difference in the effectiveness of the PDCPs. Therefore, identifying and modeling interactions capabilities of each unit type (i.e., each role) is mandatory to design effective human-machine systems.

Figure 1.2.a shows two HMUs, particularly a nurse and a physician, using a mobile collaborative application to perform hospital work. The architecture of these applications usually involves three main components (Rodríguez-Covili et al., 2011): a shared workspace (or collaboration environment), coordination services, and interaction services.

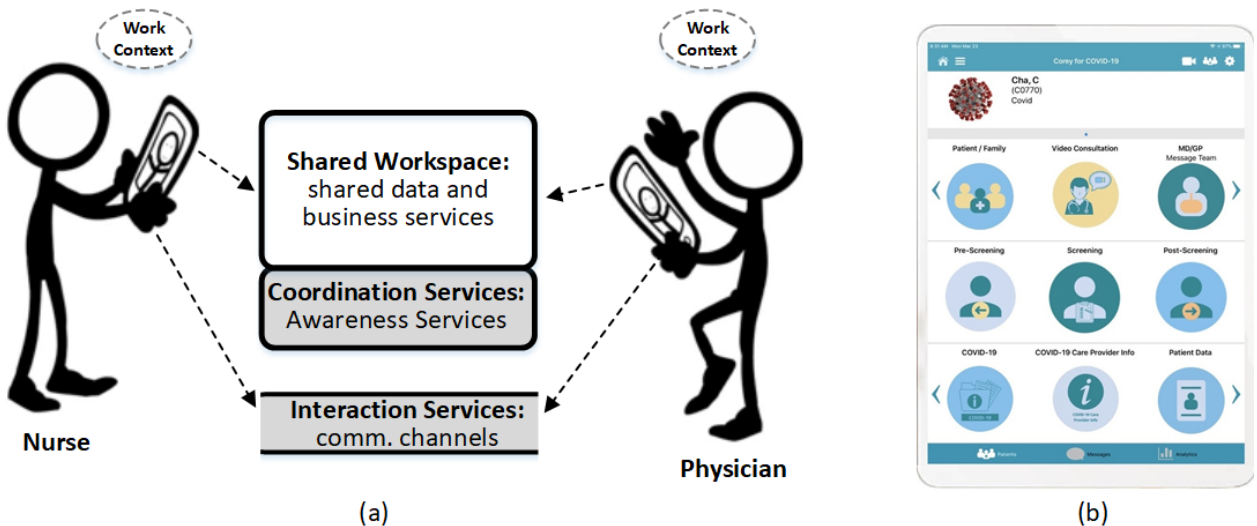


Figure 1.2. (a) Structure of a mobile collaborative application that supports the work of two HMUs, (b) workspace example of a healthcare application.

In the system architecture, the higher layer provides the user interface and allows users to consume business services. For instance, Figure 1.2.b shows a workspace of a

mobile application that allows medical personnel to monitor the evolution of Covid-19 hospitalized patients.

The second layer corresponds to coordination services that allow the system to provide visual awareness to the users; for instance, information about actions performed by other participants, or the status of the joint work and the user's context (indicated as “work context” in Figure 1.2). This layer usually implements the workflow of activities that participants perform to reach group goals. However, in the case of a PDCP, such a workflow cannot be pre-established at the application design time; therefore, only requirements of awareness services can be identified at this layer during the system design time.

Finally, the lower layer provides the services that allow HMUs to exchange files, messages and also carry out synchronous communication. Usually, these services implement several communication channels that facilitate the interaction among HMUs.

Regardless of the type of PDCP and business domain to be supported by the mobile application, the design of the interaction and awareness services (i.e., the two lower layers of the architecture) has a direct impact on the effectiveness of these systems. Although there is not a clear recipe to address this design activity, the research community tends to prefer the use of co-design approaches to deal with it (Canché & Ochoa, 2019). These approaches consider the joint work of developers and stakeholders, to determine the interaction scenarios and services of the system at analysis and design time.

1.2. Conceiving Mobile Collaborative Applications to Support PDCPs

As in any software development project, the early definitions performed on the product scope condition the solution and also its potential impact when it is used in the application domain. In case of bespoke projects where a mobile collaborative system is going to be developed to support a PDCPs, the product scope definition starts determining the services of the two lower layers of the system architecture (shown in Figure 1.2.a); i.e., the interaction and awareness services, since most of the services in the upper layer will depend on the former.

Figure 1.3 shows a typical process to address this interaction design (Canché et al., 2022b). In the first stage, stakeholders and requirements engineers agree on the interaction scenarios to be supported. This includes the identification of roles participating in the PDCP (i.e., the type of HMUs to be considered) and the interactions among them. The second stage involves the joint work between stakeholders and the provider (usually, requirements engineers) to characterize the roles and the interactions

between every pair of user roles. In the third stage, the participants decide the awareness and interaction services the system should provide to every HMU type.

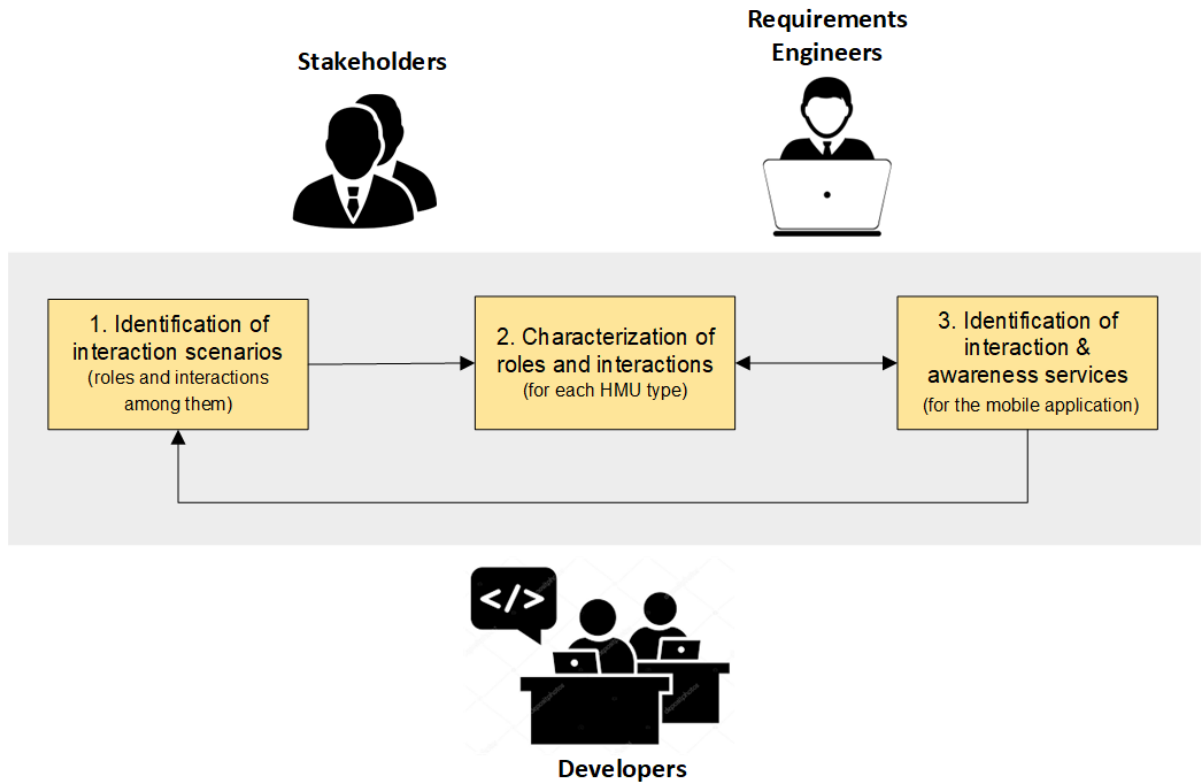


Figure 1.3. Interaction design process of mobile collaborative systems that support PDCPs.

The interaction scenarios to be supported, and also the services to be provided, typically are ad hoc to each PDCP. In other words, the supporting system should consider the particularities of the PDCP, more than the inverse. This usually happens when a PDCP represents a process already adopted by the target organization; e.g., a hospital or firefighting company. In these cases, which are highly frequent, we are talking about bespoke projects. Therefore, the resulting products (i.e., the mobile collaborative systems) will be as diverse as the PDCPs.

The interactions and services considered in these systems must be specified in a visual representation that can be understood and validated by all parties. To do that, the current modeling languages and notations allow specifying mainly the roles of participants in a PDCP, and also the interactions among them. Canché et al. (2022b) report that these languages and notations must deal with various modeling challenges, mainly due to the diversity of people involved in the conception and validation of such a visual representation.

1.3. Problem Definition

As previously mentioned, several modeling languages and notations, and also extensions to the existing ones, have been proposed to represent the interaction scenarios that are present in PDCPs. These representations make explicit the roles of the actors participating in the process, and also the interaction and awareness services required by these roles to collaborate among them.

Although these visual languages and notations have contributed to specify the interaction scenarios, their nomenclature usually does not allow dealing with the modeling challenges reported by the literature for this type of representations. Particularly, these representations should:

1. *Specify and characterize roles participating in a PDCP, and also interactions between them.* (Zhu & Zhou, 2006) indicates that specifying and characterizing these roles and interactions allow designers to build, validate and improve these scenarios, and also identify the interaction services required to support them.
2. *Be understandable for stakeholders.* Usually, the interpretation of technical representations for non-experts is slower, error-prone and it requires more conscious effort, as opposed to technical experts (Moody, 2009). While it might seem reasonable to design a notation to be understandable to stakeholders and use it also to communicate designs to developers (following the “lowest common denominator” principle), the cognitive load theory suggests that this may be incorrect (Sweller, 1994). Particularly, optimizing representations for non-experts can reduce their effectiveness for experts and vice versa; this is called the *expertise reversal effect* (Kalyuga et al., 2003).
3. *Ease the building of a shared understanding between stakeholders and developers about the scenario to be supported.* Building a shared understanding between stakeholders and software engineers is a crucial prerequisite to successfully develop and deploy almost any software system (Hoffmann et al., 2013). Typically, visual notations are used to represent and evolve this shared understanding, but these representations do not always allow people involved in the development to build the same understandings or assign the same meaning to the specification. Frequently, the misunderstandings are related to properties of the language used to represent the shared understanding; for instance its *semantic transparency*, which is the extent to which the meaning of a symbol can be inferred from its appearance; its *perceptual discriminability*, that requires that symbols should be different from each other; and its *complexity management*, which refers to the ability to represent information without overloading the human mind, among others (Moody, 2009). Therefore, these aspects should be considered by any language used to represent the shared understanding among developers and stakeholders.

4. *Allow the representation of interactions at different abstraction levels.* *Cognitive fit* (which is related to problem solving performance) is the result of a three-way interaction among the *problem representation*, *task characteristics*, and *problem solver skills*. Therefore, the modeling language and the specification process should consider these aspects when representing and validating the interaction scenario. The *cognitive fit theory* (Vessey, 1991) states that different representations of information are suitable for different tasks and audiences. However, these differences are rarely considered in the design of software engineering visual notations, even though they are routinely used to communicate and validate designs with business stakeholders (Moody, 2009).
5. *Manage the complexity and size of the interaction processes specifications.* The interaction scenarios can be large and complex, leading to create large and complex models that usually are difficult to specify, analyze and validate. Therefore, the visual notations proposed to represent these interaction models should provide mechanisms to manage this complexity; e.g., modularizing and/or using hierarchical structures in these representations. This allows participants in the modeling process to focus on particular sub-processes (or model sub-domains) without losing sight of the whole process specification. According to *ontological theory* (Weber et al., 1997), complexity management mechanisms are essential for software engineering notations.
6. *Provide mechanisms for automatic model consistency checking.* Verifying the consistency of the interaction models is mandatory to then obtain an appropriate set of interaction requirements that helps perform the collaborative system design. This activity could be done manually by developers when the model is small, but such an approach does not scale, is time-consuming, expensive and error prone. In this sense, the automatic verification of the models typically allows addressing these limitations (Mendling, 2009).
7. *Derive software requirements from the interaction models in an automatic way.* Research has shown that many projects fail because of inadequate requirements, showing that errors made in the requirements engineering stage “are among the most difficult to detect and the most expensive to correct” (Boehm, 2002). Providing mechanisms that lead to a systematic derivation of software requirements from the interaction models helps decrease both the derivation effort and human errors, while making this process less dependent on human skills (Turkman & Taweel, 2019). The software requirements to derive are mainly those potentially useful to support interactions among participants. Using this input and the agreement with the stakeholders, the analysts define the actual set of requirements to be considered in the design of the collaborative system.

Over the last two decades, several interesting proposals have emerged trying to represent these interaction models; however, they partially address the presented modeling challenges (Canché et al., 2022c). The understandability of these

representations for stakeholders seems to be the challenge that is most difficult to overcome. Unfortunately, not properly addressing such a challenge jeopardizes the possibility to reach a shared understanding among both counterparts (i.e., software engineers and stakeholders).

1.4. Work Hypotheses

In the context of elicitation, specification, analysis and validation of requirements for mobile collaborative applications, this thesis hypothesizes that:

H1) A domain-specific modeling language (DSML) allows specifying the role-based interaction scenarios between HMUs in people-driven collaborative processes (PDCPs) in a comprehensible way.

H2) Models specified in the DSML, along with prototypes automatically derived from these models, allow software engineers and stakeholders to reach a shared understanding on the interaction requirements to support PDCPs.

1.5. Research Goals

In the context of the requirements engineering of bespoke projects for developing mobile collaborative applications to support people-driven collaborative processes, the main goal of this thesis is the following:

to develop a machine-processable visual modeling language that allows developers and stakeholders to specify, analyze and validate interaction and awareness services to be embedded in mobile collaborative applications that support people-driven collaborative processes.

The use of this language, named *CIMoL (Computer-Mediated Interactions Modeling Language)*, complements the current requirements engineering techniques; particularly, for performing product scoping during the system's conception, either during pre-contract or at project start time. To address the general goal, we have defined the following specific goals:

1. Develop a visual modeling language to represent PDCPs that helps software engineers and stakeholders:
 - a. manage the complexity and size of a PDCP structure specification, and particularly, the diversity of interaction scenarios that can be embedded in these processes. It should also allow them to build a shared understanding on the PDCP to be supported.
 - b. specify interaction scenarios that can be present in a PDCP. It allows representing roles (types of HMUs) and interaction among them. Moreover, it will help both parties create a shared understanding on the

interaction scenarios being specified, and also on the communication and awareness services required by the HMUs to perform the PDCP.

2. Develop a software tool that allows software engineers and stakeholders to visually model both a PDCP and its interaction scenarios using CIMoL. The tool, named CIMoL Modeler, verifies the models' consistency during the specification of the PDCP and interaction scenarios, providing automatic feedback to software engineers and stakeholders.
3. Develop a software service that uses an interaction model as input, and automatically generates a prototype of the collaborative system to be developed. Such a prototype should consider only the roles and interactions represented in the model used as input. Moreover, it helps software engineers and stakeholders to reach a shared understanding about the scope of the product being developed; i.e., the mobile collaborative system. This prototype generation service extends CIMoL Modeler.

1.6. Research Methodology

To achieve the thesis goals, we used a methodology that involved four main stages. Each stage considered a set of activities that were not necessarily performed in a sequential manner. Some of these activities were conducted in an iterative or evolving way, to improve (through various refinements) the artifacts committed in the plan. The four main stages of the methodology were the following:

1. *Analysis of the state-of-the-art.* This stage considered the review of the literature about: (a) the structuring of collaborative processes, and particularly people-driven ones; (b) the methodologies available to guide the development of collaborative systems supporting these processes, including those that allow addressing the product scoping; (c) the techniques and methods used for eliciting people-driven collaborative processes, and building a shared understanding between stakeholders and software engineers; (d) the modeling languages and notations for specifying interaction scenarios involved in PDCPs; and (e) the design and validation aspects of visual modeling notations.
2. *Definition and formalization of the modeling language.* The main goal of this stage was to conceive a visual modeling language to specify interaction scenarios for a PDCP. For that, the following major activities were considered: (a) define a visual dialect of the modeling language for building a shared understanding between stakeholders and developers; (b) define a visual dialect for building a shared understanding among developers; and (c) define the language constructs that allows structuring a process in terms of phases and work ambits.
3. *Development of the modeling tool that eases the use of CIMoL.* The main goal of this stage was to develop a modeling supporting tool, which also allows generating prototypes of the collaborative system being developed, based on the

interaction models specified in a PDCP. This tool eases the use of CIMoL by stakeholders and developers. Particularly, this stage of the work considered the following major activities: (a) develop a software service to model interaction scenarios using the defined language, (b) extend the software tool to automatically check the consistency of the models, (c) allow developers and stakeholders select the interaction and awareness services that could be required by the HMUs to support interactions in particular scenarios, and (d) develop a generator of system prototypes to analyze and validate the user interfaces utilized by the HMUs.

4. *Evaluation of CIMoL and CIMoL Modeler.* We used two studies in order to validate the research hypotheses. First, we conducted an expert judgment study to evaluate the concrete syntax of the language, and second, we designed and performed a case study to validate the suitability of the language to support the conception and design of these systems. This last activity involved real developers and stakeholders; therefore, the obtained results represent the point of view of the practitioners.

1.7. Document Structure

This thesis document is organized as follows:

Chapter 2 presents the background that includes several aspects, like the structuring levels of collaborative processes (and particularly PDCPs), the architecture of mobile applications that supports interactions in PDCPs, the strategies to elicit requirements and develop these systems, and the process of building a shared understanding between stakeholders and software engineers during the conception of these software applications. It also introduces concepts related to the design of effective visual modeling languages and models consistency validation.

Chapter 3 reports an expert study on the capabilities of existing modeling languages, considering the challenges reported in Section 1.3. The evaluation of each language was conducted by experts; most of them were the authors of the evaluated languages and notations. As a result, we obtained a comparison table that illustrates the capabilities and limitations of each language to address the already mentioned modeling challenges. The table also shows the opportunities of CIMoL to deal with modeling aspects not addressed by the current languages.

Chapter 4 presents the foundation of the proposed modeling language (i.e., CIMoL), which was designed to model interactions among roles in scenarios involved in people-driven collaborative processes (PDCPs). Particularly, it presents the main structure, nomenclature and design decisions of the language, and also the way to use it to create the interaction models involved in PDCPs.

Chapter 5 presents an expert study designed to evaluate the suitability of the CIMoL's concrete syntax, and the capability of the language to specify interaction scenarios involved in PDCPs. In the study participated 8 researchers, experienced in modeling collaborative processes, who belong to two foreign universities; all of them having no relationship with the development of CIMoL.

Chapter 6 describes CIMoL Modeler, a software tool developed to ease the use of this language by software engineers and stakeholders. The chapter includes the requirements of this modeling tool, the manner in which the CIMoL Modeler should be used, as a complement of requirements elicitation techniques. Particularly, it indicates where the language should be used, who should utilize it, and how it should be used.

Chapter 7 presents a case study that evaluates the capabilities of the proposed language and tool when used by real software engineers and stakeholders. In the case study, the participants used CIMoL and CIMoL Modeler to define and agree the interaction support to be provided by a mobile collaborative application used by formal and informal caregivers of elderly people; i.e., to assist the participants in this particular PDCP. The study involved real software practitioners and also representatives of families that require the mobile application.

Chapter 8 presents the conclusions of this thesis work, as well as some directions for future studies.

Chapter 2. Background

This chapter addresses the theoretical aspects that are part of the background of this research, and Figure 2.1 shows a concept map with the topics addressed in the next sections.

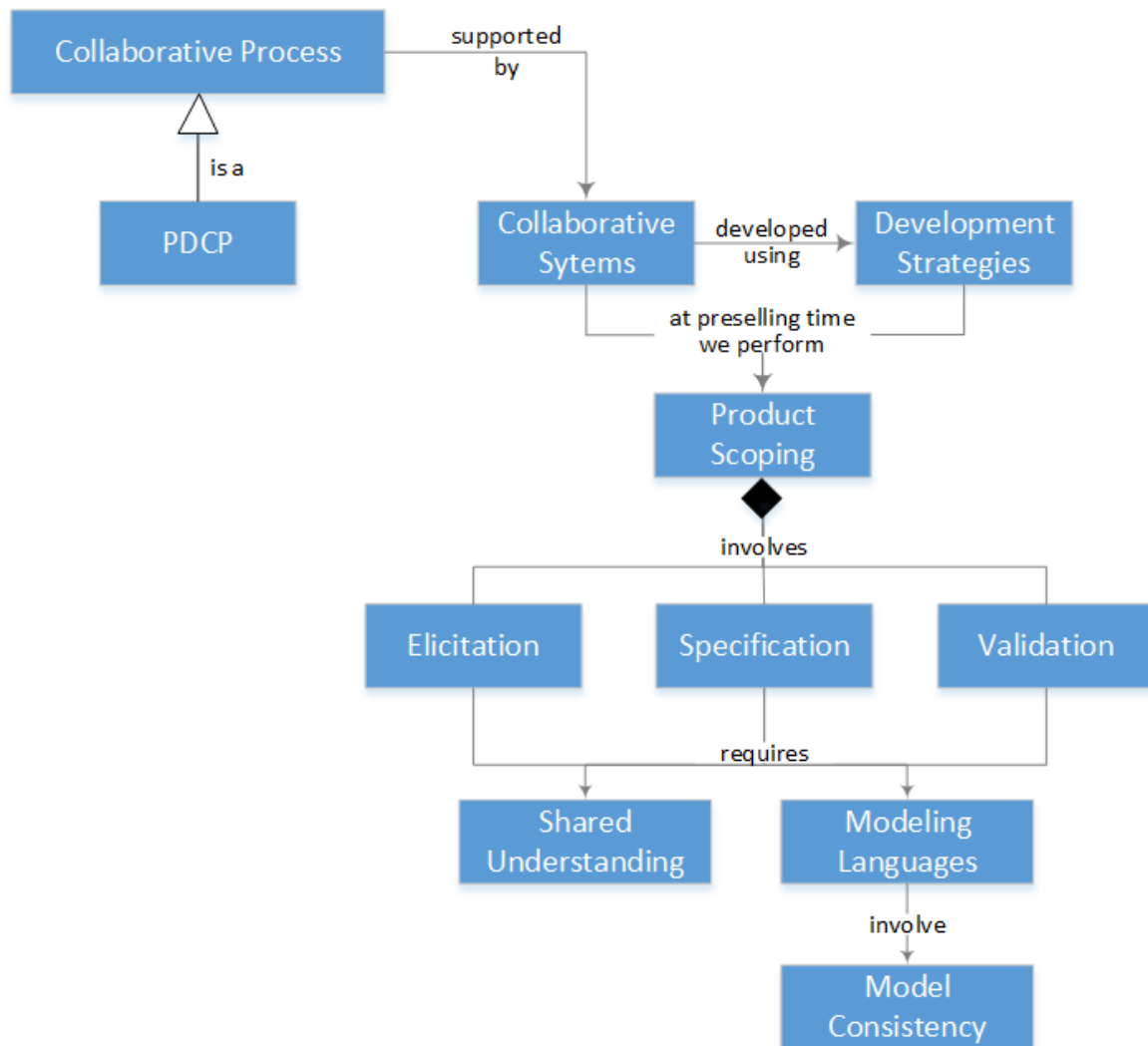


Figure 2.1. Concepts map of the topics presented in this chapter.

Particularly, Section 2.1 introduces the collaborative processes and their classification according to their level of structuring. The PDCPs are one of these classes of processes, and also the focus of this thesis work. Section 2.2 explains the main structure of the mobile collaborative applications that support the participants in these processes. Section 2.3 analyzes the main strategies reported in the literature to address the development of these systems. Section 2.4 explains how to conceive (i.e., get a preliminary design of) these systems at preselling time; i.e., to perform the product scoping. The product scoping is an early requirements engineering activity (Vera et al.,

2021b) that involves elicitation, specification and validation of the scope of the product to be developed.

Section 2.5 presents techniques and methods used to conduct the elicitation in these projects, and the way in which they contribute to both, define the product scoping and achieve a shared understanding among engineers and other stakeholders. Section 2.6 presents concepts involved in the building of a shared understanding among stakeholders and engineers, which is an important element of this study. These concepts include a classification of shared understanding and useful practices to achieve such an understanding between stakeholders and software engineers. Moreover, concepts and principles about the design of effective visual modeling languages are described in Section 2.7.

Section 2.8 analyzes several aspects of model consistency which are required to address the automatic validation of the systems and interaction scenarios specification. Finally, Section 2.9 presents a summary of the chapter.

2.1. Collaborative Processes Structuring

Collaborative processes can be classified according to their level of structuring, ranging from tightly framed (structured) to fully unframed (unstructured) processes, as shown in Figure 2.2. Next, each class is described according to the definitions presented in (Cardoso et al., 2016) and (Huth et al., 2001).

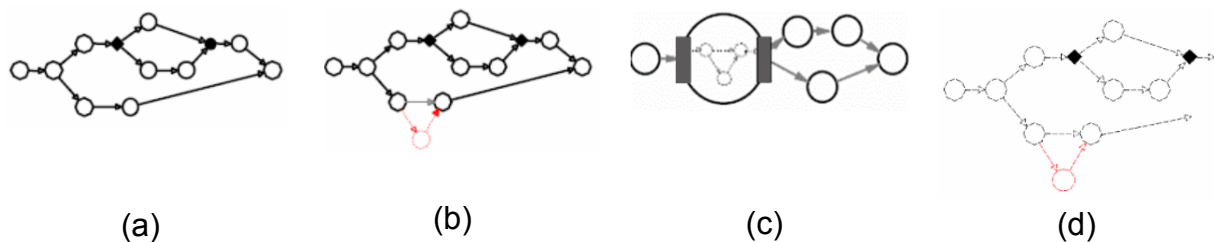


Figure 2.2. Structuredness of collaborative processes (Huth et al., 2001): (a) tightly framed processes; (b) loosely framed processes; (c) ad hoc framed processes; (d) fully unframed processes.

- *Tightly framed processes.* These processes are well structured, fully predictable and highly repetitive. In this type of processes, the activities performed by the participants as well as the activities workflow are completely predefined. These workflows pass through the same predetermined order of steps over and over again, very often it consists of routine activities. Figure 2.2.(a) shows the structure of these processes; typical examples of them are found in production and administrative processes.

- *Loosely framed processes.* These processes capture standard ways of performing tasks (Fig. 2.2.(b)). They have a completely predetermined workflow with some exceptions. An example of a loosely framed process is the processing of consumer credit applications, where there is a clear workflow, but there are also some exceptions depending on the particular customer request or customer type submitting the application.
- *Ad hoc framed processes.* These processes capture structured fragments that are then composed on a per-case basis (Fig. 2.2.(c)). Processes in this category contain a predefined workflow, but also parts of them being planned and executed in an ad hoc manner. Examples of ad hoc framed processes are agile and global software development. In this scenario, a standard guide to developing the solution for most software problems can be pre-established, however, some of them have to be planned and solved on-the-fly.
- *Fully unframed processes.* These processes are fully unpredictable and highly non-repetitive. That is, the activities performed by the participants, as well as their workflow cannot be predefined at design time. This kind of process frequently involves urgent, short-lived, exceptional, and/or confidential activities, which are typically performed considering participants' roles. Figure 2.2.(d) represents this type of process; examples of them are hospital work and disaster relief efforts.

The two latter kinds of processes (ad hoc framed and fully unframed processes) are usually also known as people-driven (Dorn & Dustdar, 2011) or knowledge intensive (Di Ciccio et al., 2015). The main characteristics of a people-driven collaborative process (PDCP) is the presence of roles played by the participants, and a lack of a pre-established workflow (or part of this workflow) that coordinates the actions of the participants. As mentioned in the introduction, the participants decide on-the-fly how and when to conduct the coordination activities, by considering several context variables, such as, the content of their to-do list, or the priority/urgency of their pending activities. The modeling of interactions in this type of processes is the focus of this thesis work.

2.2. Mobile Collaborative Systems that Support PDCPs

A collaborative process involves people (users) playing predefined roles and interacting among them to reach common goals (usually, group or organizational goals) (C. A. Ellis et al., 1991; Smith et al., 1998). These processes are commonly supported by several types of software applications, e.g., cloud-to-mobile collaborative systems (Deng et al., 2016), as a way to make people's participation flexible, and improve the efficiency of the processes.

The design of these applications involves four major design dimensions, according to the 3C Model (Fig. 2.3): *communication*, *coordination*, *collaboration*, and *awareness*. The *communication* dimension identifies the channels and mechanisms required to

support interactions among the participants in a PDCP. This dimension also determines the basic services required to *coordinate* people’s activities, and integrate and share the interim results. The *communication* and *coordination* services are usually required to allow *collaboration* among the participants, and thus help them reach their individual and group goals.

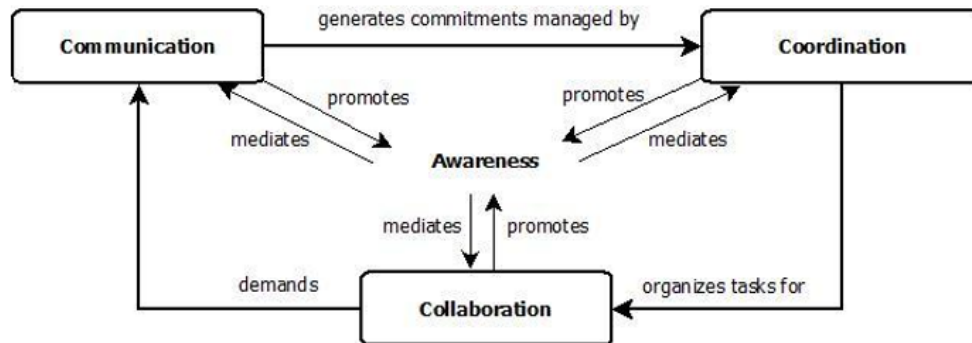


Figure 2.3. The 3C Model, adapted from (C. Ellis & Wainer, 1994; Fuks et al., 2008).

The *awareness* dimension includes the mechanisms (usually specified through visual representations) that keep participants informed about the joint work, and mediate the communication, coordination, and collaboration among these people. Coordination, collaboration, and awareness are design aspects that depend on the communication support provided by the PDCP. Designing the communication aspect of a collaborative system (i.e., the interaction services) is mandatory to then address the other design aspects.

2.2.1. Using Mobile Collaborative Applications in PDCPs

As mentioned, we use the term “human-machine unit” (HMU) to refer to a mobile worker that uses a collaborative application to participate in a PDCP. Typically, these HMUs perform loosely-coupled work (Churchill & Wakeford, 2002), i.e., the HMUs work autonomously most of the time and carry out sporadic on-demand collaboration processes. Once finished the collaboration activity, these units return to autonomous work (Pinelle, 2004; Pinelle & Gutwin, 2005).

Figure 2.4 shows two HMUs interacting; they play the role A and B respectively, and use a shared workspace (e.g., the mobile app of Uber) to participate in a PDCP. For instance, let us suppose that a sales company contracts Uber to transport their salesmen with minimum delay among several places they have to visit during a working day. In this case, the role A could be the salesmen and the role B can be a particular subset of Uber drivers. Both roles can use a particular flavor of the Uber application to coordinate the pick-up and drop-off times, monitor availability of the counterpart, or estimate trips duration.

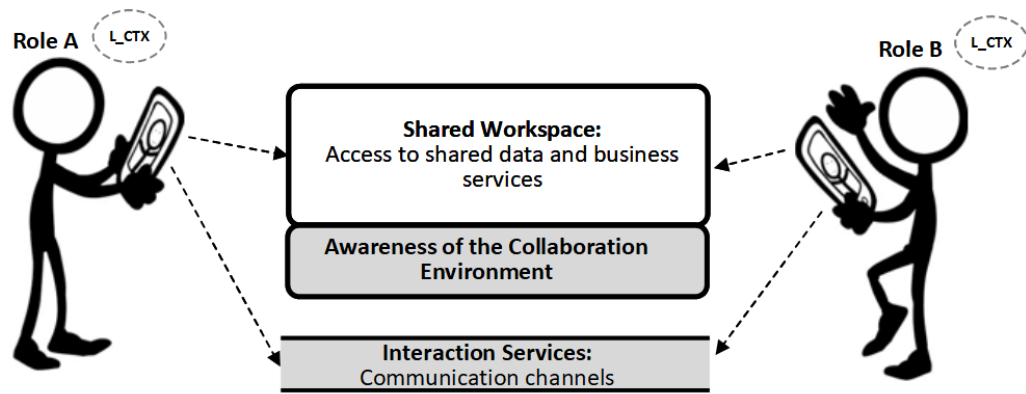


Figure 2.4. HMUs interacting through a mobile collaborative application.

The front-end of the application usually represents the major part of the shared workspace; therefore, it changes with every application. The information and services available to the user, and also the way to visualize them, are ad hoc to the PDCP being addressed by these HMUs.



Figure 2.5. Interface of the Uber workspace.

Figure 2.5 shows the Uber workspace. It embeds awareness mechanisms that allow drivers and potential passengers to know what is happening in the collaborative environment (e.g., what cars are in the area, or where the current location of the assigned driver is).

In this sense, the application usually provides contextual information to HMUs (usually as visual awareness) about the collaboration environment. However, each HMU has information of its own local context (L_CTX in Fig. 2.4), which is kept in the users' minds. The HMUs use the information of both contexts to determine their next action, therefore, their workflow is defined on-the-fly. The HMUs participating in a PDCP perform several activities depending on their own needs, or address the requests of other units. In such a dynamics, it is quite frequent that an HMU performs autonomous work, i.e., it interacts only with the system.

When the HMU needs to interact with others, e.g., to perform ad hoc coordination activities, the collaborative system should provide them: 1) particular services that support the interaction between every pair of roles, and 2) awareness information that allows the HMUs to determine when and with whom to interact. Therefore, the design of the interaction support to be embedded in the mobile application must identify these services.

Typically, it is done as part of the requirements engineering and preliminary design stages. Consequently, the languages used to model interaction scenarios should allow representing these services, since they are part of the agreement and shared understanding between the stakeholders and provider.

2.2.2. Types of Participants in a PDCP

In order to simplify the explanation, the previous sections present examples involving only human actors as users of mobile applications. However, several types of actors can participate in the interaction scenarios, and the visual languages used to represent them should allow specifying the type of the participants. Figure 2.6 shows a more complete interaction scenario, where we can identify internal and external users. The first ones are actors (end-users) that utilize the application being developed, as an instrument to interact with others. Each actor type (i.e., users playing a specific role) can access a particular functionality of the mobile application, based on a schema of grants.

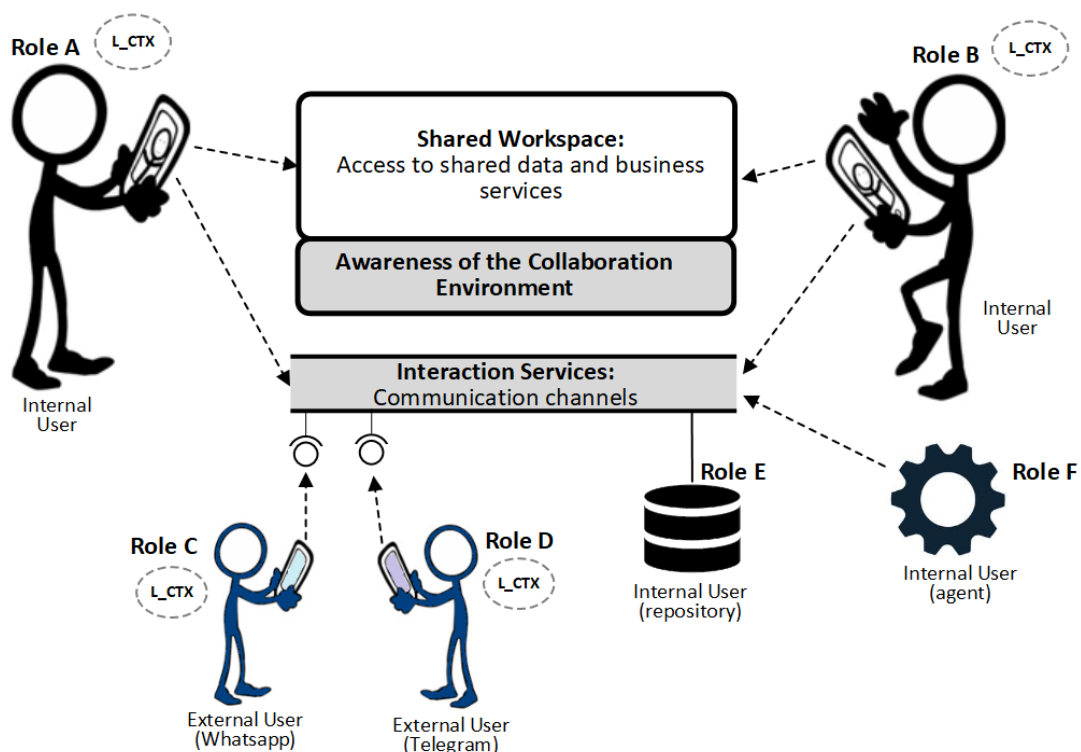


Figure 2.6. Types of participants in interaction scenarios involved in PDCPs.

The internal users can be human beings, but also information repositories or software agents. The repositories are passive actors that act as intermediary between other participants. The software agents are autonomous components that have a set of services predefined to interact with others. It is assumed that repositories and software agents have high availability to perform interactions; in this sense, their uptime is considered similar to a regular cloud service.

As shown in Figure 2.6, agents and repositories do not have a local context as the human being. Their local context, if it exists, is part of the shared workspace.

On the other hand, the external users are human actors that interact with others using a system that is different to the mobile application being developed. For instance, a user can utilize Telegram or Whatsapp applications to communicate with actors that use the ad hoc mobile collaborative application. Having external users allows the participation of people that are not willing to install the ad hoc collaborative application. For instance, a physician (i.e., a particular mobile worker) can participate in several PDCPs using Whatsapp, regardless of whether other users utilize an ad hoc application to interact with him/her. When external users are going to participate in PDCPs, the collaborative system should provide a software interface (usually an API) with external applications to allow the interaction with those users.

2.3. Strategies for Developing Mobile Collaborative Systems

Despite historically the development of collaborative systems has been carried out in an ad hoc manner, given its particular nature, the literature shows that two types of development strategies are predominant (Canché & Ochoa, 2019). The first one focuses on system design (mainly in HCI and collaborative processes design), and the other on systems engineering (formal lifecycles). Additionally, there is a third strategy that mixes the previous approaches to mitigate weaknesses and enhance strengths. Below we discuss the most representative strategies of these three approaches, and the manner in which these strategies contribute to defining the product scoping in terms of interaction services to be embedded in a mobile collaborative application.

2.3.1. Development Focused on the System Design

The design-centered development involves the stakeholders (mainly, clients and users) during the design process (Baxter & Sommerville, 2011). Methods using this approach represent a design philosophy more than a software development methodology, because they describe a process and a humanistic set of principles related to both, the technology design and the impact that its introduction would produce in the involved people (Mumford, 2006). Among the most relevant design-centered proposals to support the development of collaborative systems we have the following:

- *Evolving model.* (Tang, 1991) defined an evolving model that allows designing collaborative systems conducting three activities (or stages): observing the activity to be supported, understanding it, and designing a system to support it. Similarly, (Nolte & Prilla, 2013) proposed a life-cycle to develop collaborative systems, which is based on evolutions of a system prototype. Figure 2.7 shows both evolving proposals. An important advantage of these approaches is that several cycles are conducted to design the system in an evolving way; therefore, the product scoping can be refined jointly with the stakeholders. These approaches are coarse-grained, and represent a guideline to design collaborative systems. They are not very useful for guiding developers through a systematic process. However, the fact of integrating experiences of stakeholders and developers throughout several cycles to improve the definition of the product scope, is a valuable asset.

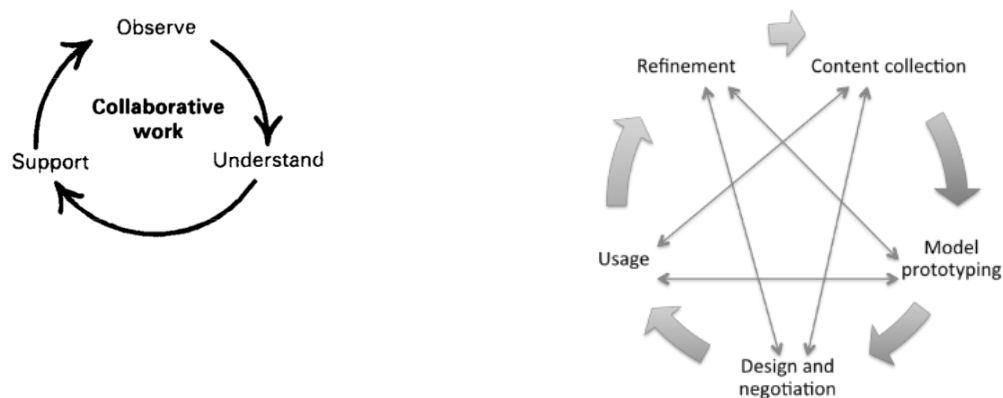


Figure 2.7. Evolving process: (a) approach proposed by (Tang, 1991) and (b) approach proposed by (Nolte & Prilla, 2013).

- *Groupware conceptual models.* (C. Ellis & Wainer, 1994) state that from the point of view of the users of a collaborative system, we can characterize such a system in three complementary aspects (or models): (1) the ontological model that describes the objects and their operations in the system, (2) the coordination model that determines the dynamic aspects of the system (e.g., floor control and data flow), and (3) the user interface model that establishes the interfaces between the system and users, and also among users. These models have strong similarities with system specification concepts used in software engineering. However, the proposal is far from useful in practice since it assumes that the collaborative work supported by the system will always be articulated by a coordination model (or component). This assumption is contrary to modern meanings of the collaboration concept that consider low and no coordination (Lee & Paine, 2015), which is a strong limitation to obtain a clear definition of the product scope.

- *Participatory design*. In this approach the users and other stakeholders are actively involved in the system design throughout the whole development process (Schuler & Namioka, 1993). Participatory design privileges users in design decisions and empowers them to co-determine their collaboration needs (i.e., the functionality to be provided by the system) (Ehrlich, 1999). In spite of the potential difficulties and costs raised by the stakeholders' participation in the whole development process, this approach, if it is conducted properly, can contribute to obtaining a good definition of the product scope.
- *Meta-design of socio-technical systems (Meta-design of STS's)*. This approach extends the design-focused methods by allowing users to become co-designers at use time (i.e., the stakeholders evaluate the product at the design and deployment time), as shown in Figure 2.8 (Fischer & Herrmann, 2015). It provides a conceptual framework to help stakeholders identify the major functionality of the system, and then its evolution considering the organizational and social changes in the environment where the system is used. Such a framework, instead of attempting to build a complete system, proposes to evolve the solutions over time through the small contributions of a large number of people. This allows the problem owners to create the solutions themselves, at use time, providing to developers a context and a background against which situated cases, coming up during using time, can be interpreted. However, given the iterative nature of this approach, the product scope definition cannot be performed before the bespoke project starts.

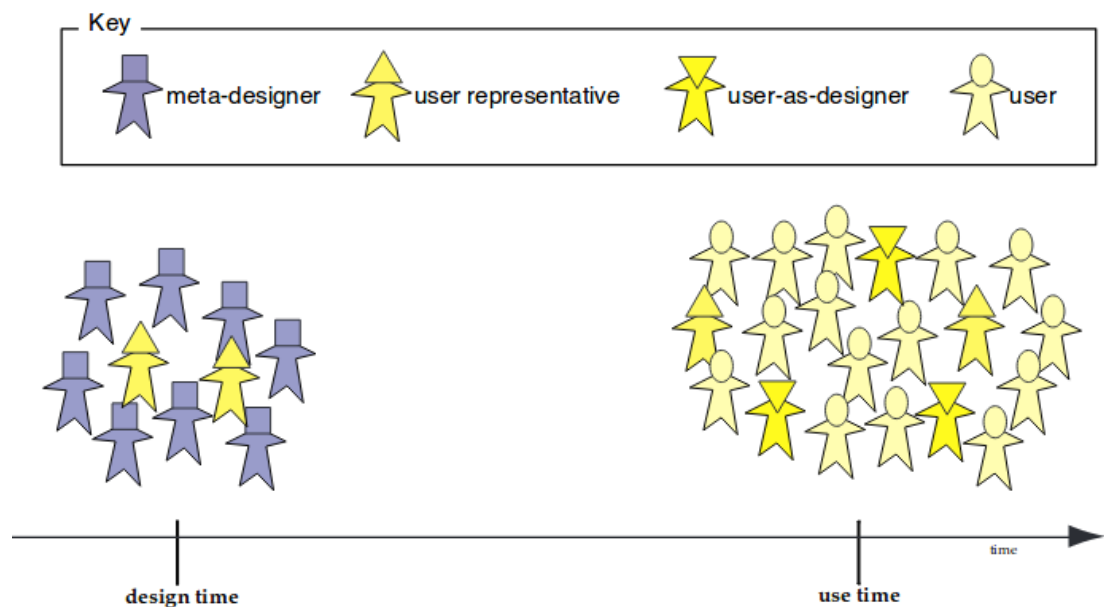


Figure 2.8. Meta-design of socio-technical systems: design and use time-roles and involvements (Fischer & Herrmann, 2015).

2.3.2. Development Focused on the Engineering of the System

The engineering-centered development of groupware systems is based on four fundamental, and not necessarily sequential, activities: *procurement*, *analysis*, *construction* and *operation*. This approach considers socio-technical factors at all stages of the system life cycle, and helps engineers and stakeholders avoid obtaining non-contextualized solutions (Baxter & Sommerville, 2011). The methodological proposals adhering to this approach are the following:

- *Component-based groupware development*. This proposal combines the component-based software development paradigm (Szyperski et al., 2002), with a conceptual cooperative model presented in (Guareis de farias et al., 2000b). This approach helps design and structure groupware applications in terms of a set of components (Guareis de farias et al., 2000a). It identifies four levels of abstraction in a groupware application: enterprise, system, component and object as shown in Figure 2.9. In order to define the product scope, the approach proposes to capture information from the system domain at the enterprise level. However, it does not address the specific way of involving stakeholders, given that their participation is optional.

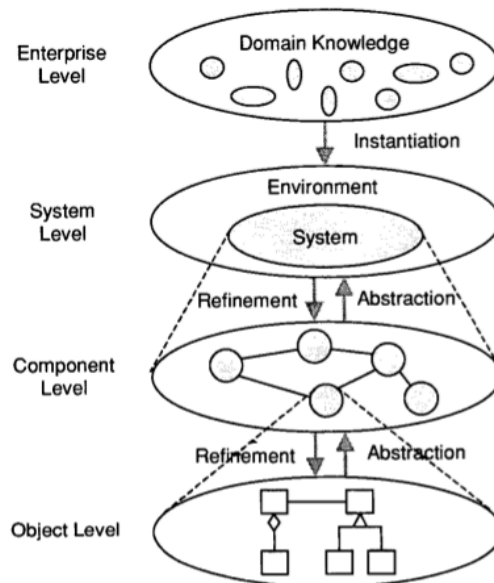


Figure 2.9. Abstraction levels of the methodology proposed by (Guareis de farias et al., 2000a).

- *System development based on roles and their interactions*. This approach (represented in Fig. 2.10) focuses on the development of collaborative services based on a study of the user roles and the potential interaction between them (Herskovic et al., 2019). Considering these roles and interactions, the model helps engineers identify the software requirements to support the collaboration process. Particularly, it covers the first two layers of the 3C model (C. Ellis &

Wainer, 1994; Fuks et al., 2008), i.e., the communication and partially the coordination between peers of users. The users play specific roles during a collaborative activity, and therefore, they perform tasks that require coordination with other team members. This proposal supports the system requirements and design phases, which could contribute to obtaining a clear product scope.



Figure 2.10. Development based on roles and their interactions by (Herskovic et al., 2019).

- *Groupware engineering.* This approach proposed by (Gerosa et al., 2005) is based on the spiral software development model, which combines the classical sequential model and the iterative behavior of incremental prototyping (Fig. 2.11). This proposal is the closest to an engineering approach for developing collaborative systems; however, it does not indicate how and where stakeholders can be involved in the process. Depending on it, it could be possible to get a good product scope definition.

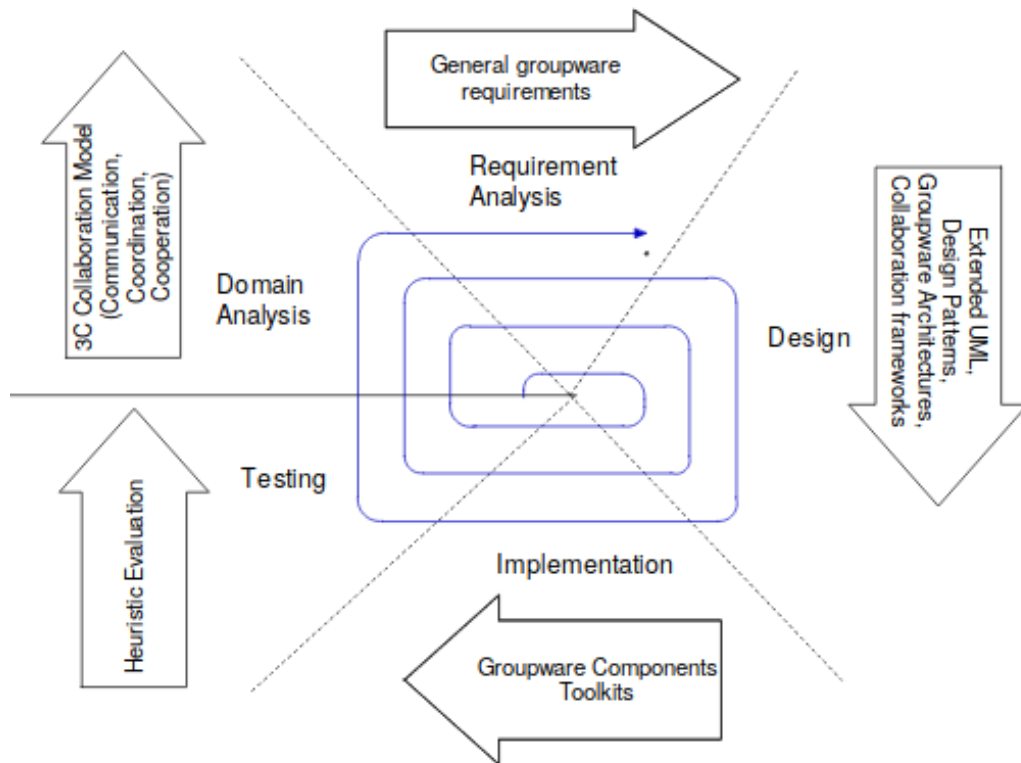


Figure 2.11. Groupware development cycle proposed by (Gerosa et al., 2005).

2.3.3. Hybrid Development Processes

Some hybrid proposals have emerged as alternatives to guide the development of collaborative applications. (Canché & Ochoa, 2019) characterize an approach as *hybrid* if it considers participatory design in the system design, uses a specific structure(s) for the development stages (some more systematic than participatory design), and uses possibly a mixture of other methods and techniques to support the development of a collaborative system. We next indicate the proposals aligned with this approach.

- *GO-SODA Framework*. This development method combines a goal-oriented approach and a technique called SODA (Strategic Options Development and Analysis) (Osguthorpe et al., 1996). This combination is instantiated through a conceptual framework that involves five organizational views or specifications (particularly: business, group, role, application, and system view) and allows for progressively describing organizational goals and requirements (Fig. 2.12). This method follows an operational research approach designed to help identify and deal with a complex set of key issues in an organization. To obtain a good definition of the product scope, this method promotes communication among stakeholders and developers during the development project. For that, small user groups are initially defined using permanent feedback concerning the various modeling techniques used to represent their requirements.

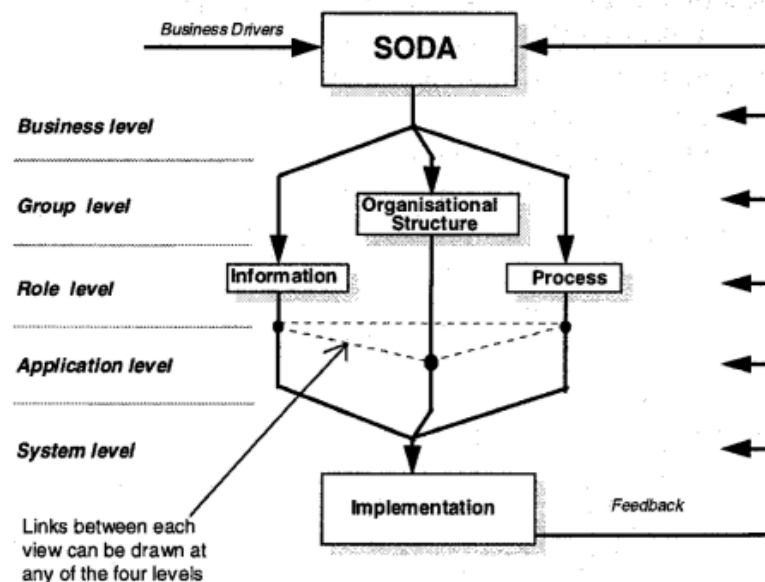


Figure 2.12. Framework proposed by (Osguthorpe et al., 1996).

- *Multi-method approach*. The multi-method approach typically uses an iterative process based on high-level functional prototypes to discover the functionality, constraints and user interfaces of the final product (Pekkola et al., 2006). This approach, which is represented in Figure 2.13, begins with a typical information

systems development lifecycle, conducting the requirements specification and domain analysis, and then performing their refinement for using them as input to the design and implementation of system prototypes. This proposal is mainly a mix between the traditional spiral model and participatory design (Schuler & Namioka, 1993). This proposal is suitable to obtain a good definition of the product scope, as long as some key conditions are met. The system prototypes are evaluated, first inside the development team and then with a small sample of the target population, to obtain improvement recommendations as well as identify possible design flaws.

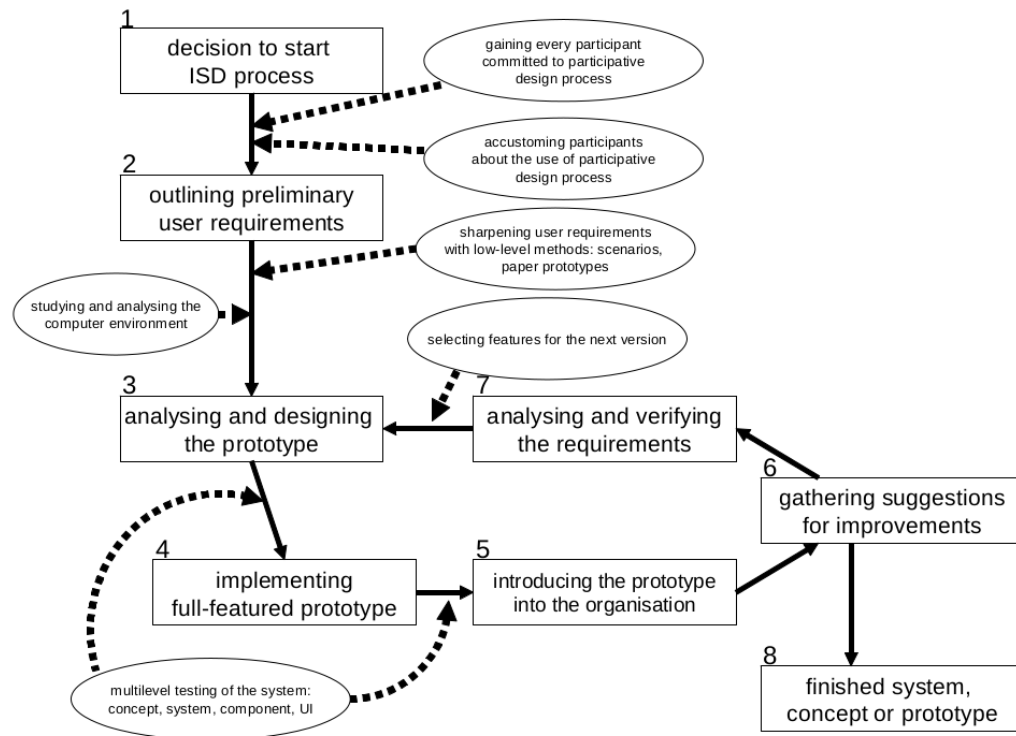


Figure 2.13. Approach proposed by (Pekkola et al., 2006).

- Framework for CSCW systems development.** This approach uses a conceptual and technological framework for supporting the development processes, and a quality evaluation method for collaborative systems (Antonaya & Santos, 2010). The structure of a collaborative system, according to this framework, is shown in Figure 2.14. The conceptual framework allows specifying the functionality of the collaborative system considering the use of high level abstraction models and dividing a full system into three independent and complementary specifications: structure, behavior, and instrumentation. Then, the quality of the resulting specifications is evaluated, and the results are used to refine the previous design through an evolving approach considering the stakeholders' participation through the whole development process. However, similar to other proposals, it has an important limitation: it does not specify how and where stakeholders can be

involved, particularly regarding the requirements engineering stage, which makes it difficult to get an adequate product scope.

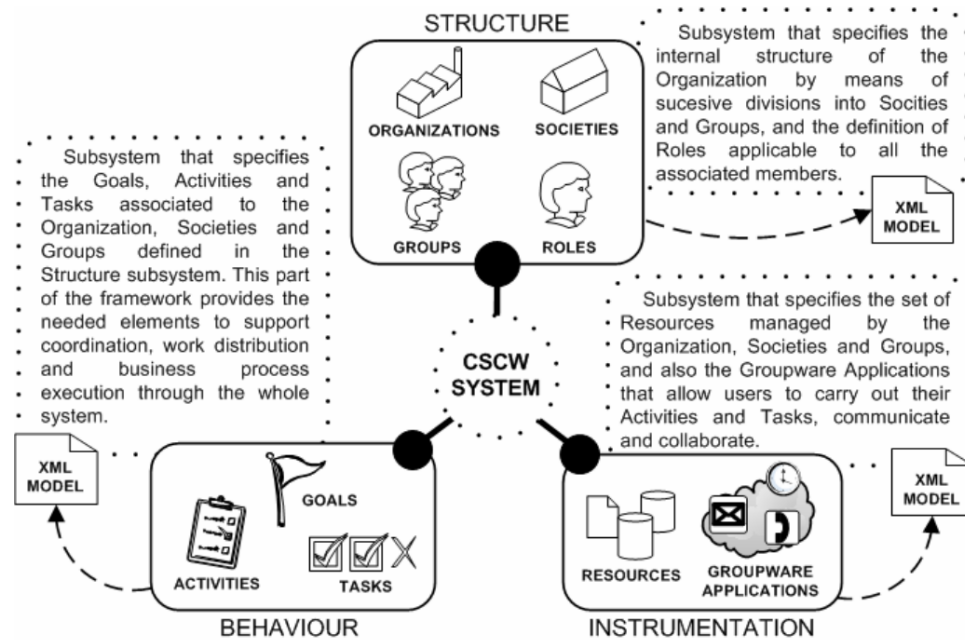


Figure 2.14. Structure of a CSCW system according to the conceptual framework of (Antonaya & Santos, 2010).

2.4. Product Scoping

The *product scope* represents the main input that allows the provider to generate a project proposal and a bid, and then deliver them to the customer (Savolainen et al., 2015; Vera et al., 2021b). We next discuss the relevance of doing a proper and early definition of the product scope, and also how to perform product scoping in bespoke projects.

2.4.1. Relevance of the Early Definition of the Product Scoping

Bespoke software projects usually start with a pre-contract stage (also known as pre-selling), where software providers interact with the customer or stakeholders to understand their needs. Based on that definition, the provider generates a project proposal that intends to be suitable for both parts (Savolainen et al., 2015). Eventually, the project proposal produces a contract, i.e., a software project (Figure 2.15).

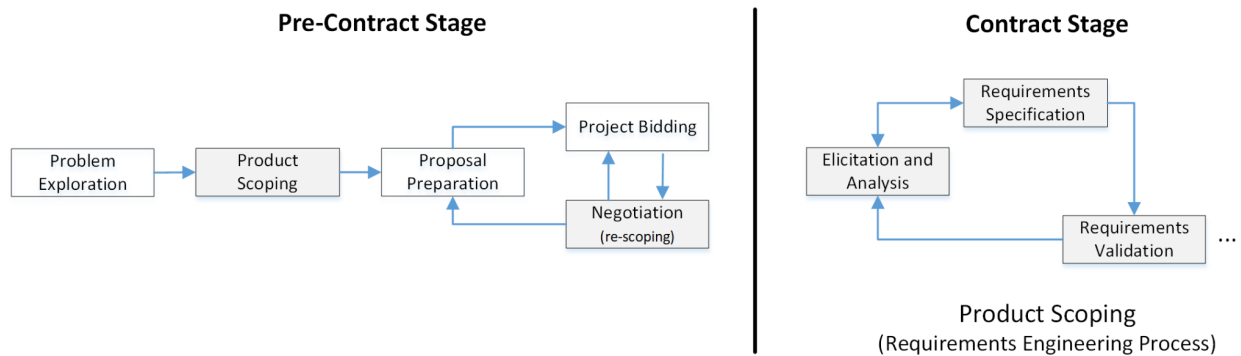


Figure 2.15. Scoping at pre-contract and contract stage (adapted from (Vera et al., 2021b) and (Savolainen et al., 2015)).

At the pre-contract stage, the customer and the provider explore the problem or opportunity to address, and then define and agree the goal and scope of the product to be bid. This second activity is known as *product scoping* (Vera et al., 2021b), and it represents the main input that allows the provider to generate a project proposal and a bid, and then deliver them to the customer (Savolainen et al., 2015; Vera et al., 2021b). Frequently, when the provider delivers a proposal, it triggers a negotiation between both parts, where the project scope (i.e., the software features to be included into the product), the project plan or the bid is adjusted generating a new proposal. If this negotiation and refinement process is successful, it usually gives place to a software development contract.

At the project start-up, the product scoping activity is performed again (this time, as part of the requirements engineering process), to determine the product features with more details, and thus to establish the development time and cost more accurately, and also the resources required to carry out the development project. Typically, the scoping process is iterative and involves the requirements engineering major activities: elicitation, analysis, specification and validation. The requirements management is part of the product scope management.

The product scoping is part of the product conception process, and it represents a key component that affects the project success (Woolridge et al., 2009). Performing an accurate definition of the product scope at an early stage of the project is mandatory, since it strongly influences the project's success or failure (Hassan & Asghar, 2021; Kumari & Pillai, 2014; Mirza et al., 2013; Taherdoost & Abolfazl, 2018). The product scope definition includes the do's and don'ts of the projects, and it represents the prospect of the project for stakeholders and the provider (Hassan & Asghar, 2021; Taherdoost & Abolfazl, 2018).

A poor product scope definition imposes a variety of adverse effects on the product suitability, and project schedule, cost, and quality (Hassan & Asghar, 2021; Lampa et al.,

2017; Mirza et al., 2013). Poor definitions usually extend the project time and development effort, and disrupt the rhythm of the development team (Komal et al., 2020; Mirza et al., 2013; Shmueli & Ronen, 2017).

2.4.2. Product Scoping in Bespoke Projects

As mentioned before, in bespoke projects the output of this phase is a shared understanding about the product goals and scope (i.e., the user requirements), which increases the chances to reach an agreement (a contract) between customer and provider.

As shown in Figure 2.16, the project leader (or requirements engineer) interacts with the stakeholders to determine, refine and validate the user requirements (UR), and then with the developers to realize how to deal with the URs; i.e., to translate the URs to software requirements (SR), and thus to determine the features of the system.

The SRs should also be specified in a representation that can be understood by the project leader and developers, as a way to ease the reaching of an agreement between them about the services that the system should provide. Similar to the URs, there should be a shared understanding about the SRs to be included into the system.

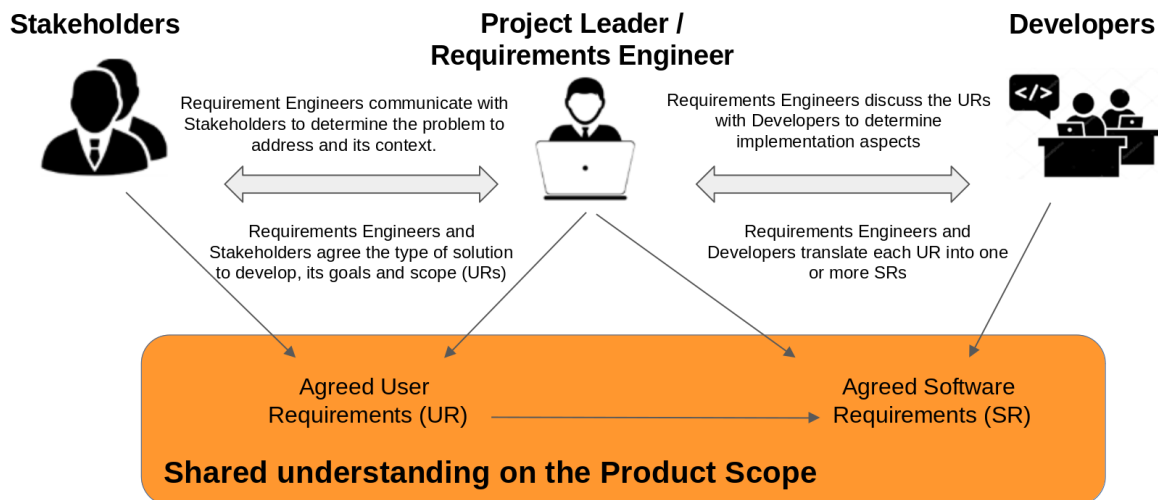


Figure 2.16. Interactions among participants in the scoping phase.

Vera et al. (2021b) report a study that explores the software engineering practices used by small companies to perform the scoping phase at pre-selling (or pre-contract) time, and also the effectiveness of those practices. The study results indicate that companies use a mix of five practices to support the activities of the scoping phase: interviews, high-level requirements specifications, analysis of legacy information, search for information on the Internet, and pilot projects. In the first four practices the role of the stakeholders is passive, and they do not produce an outcome that is easy for them to

understand; the output is mainly for technical personnel. Consequently, these practices (even when used combined) have low cost- effectiveness. The last practice (i.e., the pilot projects) produces an output that stakeholders understand, and therefore, they can help refine and validate jointly with the provider. Unfortunately, this option is rarely used due to its cost and the effort required to reach a product scope definition.

To deal with this activity in a cost-effective way, Vera et al. (2021b) propose a collaborative product scoping process (Fig. 2.17) and a particular visual representation to make explicit the product goal and scope. These representations, that are based on visual canvases, are usually understandable for stakeholders and software providers, and allows them to get a shared understanding about the product scope (Vera et al., 2021b).

The scoping process receives the customer request as input, and then the participants (i.e., stakeholders and the provider’s personnel) perform three activities through an evolving process, until getting a product scope specification that is understandable for both parts. These activities are solution discovery, product scope definition, and scope validation.

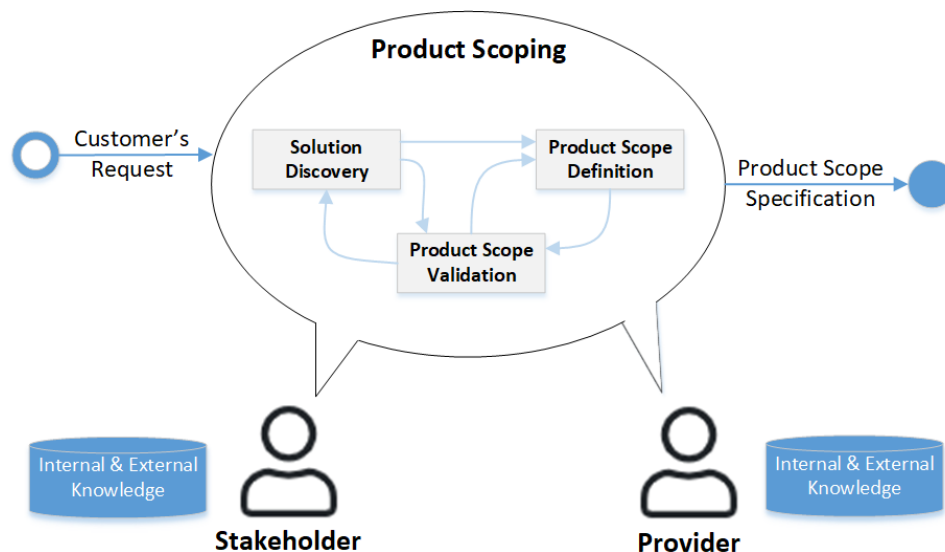


Figure 2.17. Product scoping in bespoke projects (from (Vera et al., 2021b)).

In terms of major activities, Figure 2.17 shows that there is a general process to guide the product scoping. However, performing the scope definition of diverse system types requires identifying several components and functionalities of the product; i.e., the solution usually talks about concepts that are particular from a software system type.

Usually, the main structure and functionality of a particular system type adhere to the reference architecture defined for systems that fall into such a category (Vera et al., 2021a). Collaborative systems are one of these system types that involve particular

components, which are structured in a certain (general) way. Using such a structure, these systems provide communication, coordination and collaboration services. The 3C Model (C. Ellis & Wainer, 1994; Fuks et al., 2008), explained in Section 2.2, provides a reference architecture for these systems.

2.5. Requirements Elicitation for Collaborative Systems

Any development strategy requires eliciting requirements. *Requirements elicitation* is concerned with discovering the sources of requirements (e.g., stakeholders or business processes), identifying their needs, negotiating potential conflicts, and establishing a clear scope and boundaries of the system.

Information gathered during requirements elicitation often has to be interpreted, analyzed, specified, and validated, before the requirements engineer can feel confident that a complete enough set of requirements of the system have been collected (Nuseibeh & Easterbrook, 2000). Researchers recognize that the type of the system to be developed, project purpose, and communication forms between elicitors and stakeholders significantly affect the way in which requirements elicitation is conducted (Carrizo et al., 2014; Coughlan & Macredie, 2002; Hickey & Davis, 2002; Zowghi & Coulin, 2005).

Several techniques to elicit requirements have been proposed over the years (Zowghi & Coulin, 2005). Some of them are widely used in practice, others are relatively unknown by practitioners or rather theoretical. Cost may be an important consideration when choosing a technique for the groups involved, especially in terms of time invested with each technique. A number of other factors may influence the choice of a specific technique or combination of techniques (Carrizo et al., 2014).

Concerning collaborative systems, requirements elicitation represents a major challenge for developers due to the way in which the users interact with the system and with other users, that cannot be prescribed at software design-time. In fact, the suitable way to share information collaboratively depends on the involved people.

Compared to other systems, collaborative applications are distinguished by the fact that the users are engaged in a shared goal, and have the need to interact closely with each other to reach such a goal. To support this interaction the design of collaborative systems should consider mechanisms for communication, coordination, collaboration, and awareness provision. Hence, the success of such systems depends on the quality of the definition of requirements. The quality of the requirements is greatly influenced by the techniques employed during requirements elicitation (Hickey & Davis, 2002).

Canché & Pino (2021) performed a systematic literature review on elicitation techniques, to understand which of them are typically used or recommended to be used in the development of collaborative systems. The findings of that study revealed that

interviews, observation and audio/screen recording were the most frequently used elicitation techniques. Additionally, the elicitation techniques were also grouped considering two classification perspectives. Both perspectives are discussed below including the manner in which the techniques contribute to define the product scope.

2.5.1. Perspectives of Elicitation Techniques for Collaborative Systems and their Contribution to Define the Product Scope

Figure 2.18 shows the first perspective considered in the study carried out in (Canché & Pino, 2021): *elicitation techniques according to the typical characteristics that an elicitation technique has*. Under this perspective, whereas in some categories the number of techniques is high, in others the number was quite low. It is remarkable that several studies used *collaborative techniques* to support the elicitation. This preference seems reasonable because the development of such systems requires effective and efficient collaboration among various participants.

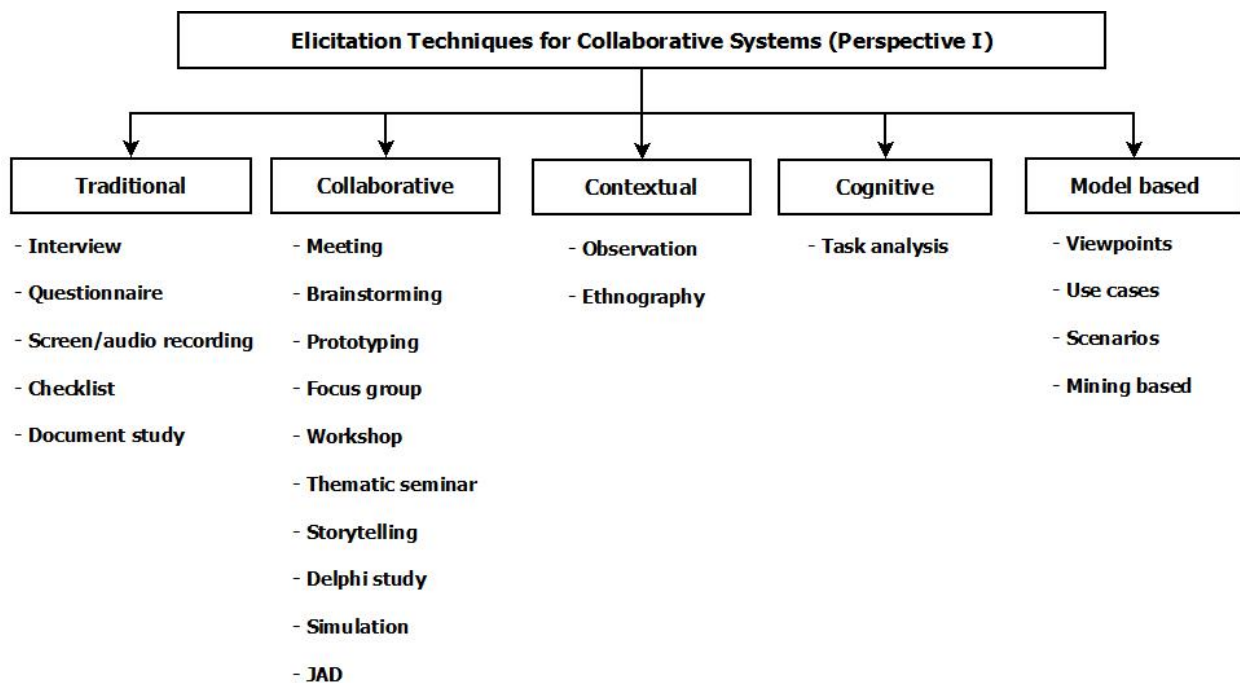


Figure 2.18. Elicitation techniques according to the typical characteristics that an elicitation technique has.

Eliciting group insights of several stakeholders in a collaborative manner is a valuable factor to derive both: suitable requirements and an adequate definition of the product scope. Therefore, these techniques, applied in an adequate manner, are effective tools in requirements engineering.

Under the second perspective described in (Canché & Pino, 2021): *elicitation techniques according to the manner in which the communication with the stakeholder is*

performed, which is shown in Figure 2.19, some interesting issues were found to be taken into account. Among them are the following:

- Most of the techniques from the “*Individual Stakeholder responses*” category were complemented with techniques from the “*Group Stakeholder responses*” category or the technique from the “*Observation of Stakeholders*” category (or with both). Given the collaborative systems nature, such a combination of techniques tends to be appropriate.
- The technique in the “*Elicitor taking role as Stakeholder*” category (i.e., ethnography) can be used effectively in combination with techniques from the remaining categories, with the exception of the “*Iterative interaction with Stakeholders*” category. This exception is because prototyping was mainly used together with the observation technique (i.e., observation was considered enough to achieve the inquiry goals) or together with interviews, questionnaires, and scenarios.

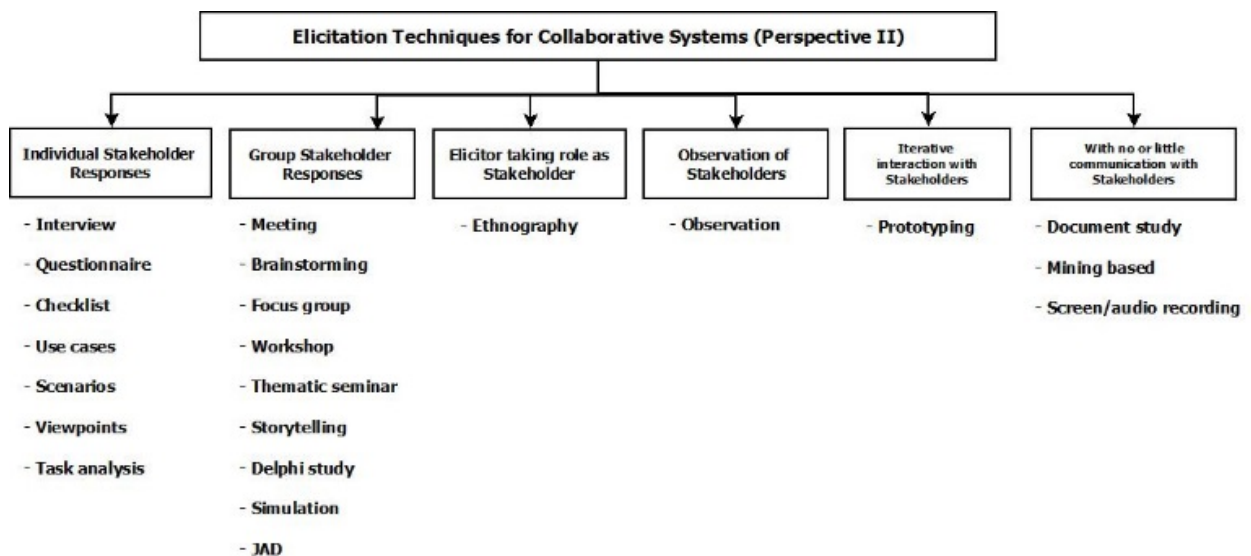


Figure 2.19. Elicitation techniques according to the manner in which the communication with the stakeholder is performed.

The results of the study show that the categories from this second perspective are not mutually exclusive. Some factors, such as expertise and communication skills of stakeholders, available resources, budget, and time among others, should be analyzed to design a suitable combination of elicitation techniques, and therefore, obtaining both: an adequate requirements specification and product scope definition.

Similar to other application domains, in the development of collaborative applications the choice of elicitation technique depends on, besides of the characteristics of the specific system, the time and resources available to the requirements engineer. Additionally,

engineers should analyze the characteristics, benefits and drawbacks of the various elicitation techniques and also assess the affinity with the strategy chosen to develop the collaborative system. For instance, if an evolving strategy is chosen to develop the system, the technique of prototyping could be a suitable option.

2.5.2. Requirements Elicitation for Collaborative Systems Supporting a PDCP

Clearly, the collaborative domain requires placing emphasis on key design aspects of collaborative systems (i.e., communication, coordination, collaboration, and awareness). Such aspects should be then considered and analyzed when elicitation techniques are selected. Moreover, when a system supporting a PDCP is developed, it should be considered that usually only aspects of communication and awareness are known. This should allow engineers to focus and take advantage of the capabilities provided by the chosen requirements elicitation technique(s). Identifying the interactions among participants of the process, and also the mentioned design aspects (particularly, communication and awareness requirements), eases engineers to provide the necessary means to perform the PDCP process.

Combining specific techniques has been a useful strategy used by practitioners to carry out an elicitation process, and it can be a valuable tool to achieve a shared understanding between stakeholders and engineers. For instance, using interviews, prototyping and observation has been effectively applied to conduct an elicitation (Canché & Pino, 2021), which could foster the shared understanding when used in the requirements engineering stage of a system supporting a people-driven collaborative process.

Regarding people-driven collaborative scenarios, due to their dynamism and unpredictability, the coordination and collaboration services cannot be defined at design time, which increases the complexity to define the whole product scope. Nevertheless, with eliciting the group goals to be reached, and also the communication and awareness services required by the participants in their interactions, an important part of the collaborative system (i.e., its scope definition) can be specified. This scope definition should be understood by all the stakeholders and engineers involved in important decisions of the project, by which a shared understanding of the requirements among both parties is necessary. Shared understanding is discussed in detail in the following section.

2.6. Shared Understanding between Stakeholders and Software Engineers

Shared understanding between stakeholders and software engineers is a crucial prerequisite for successful development and deployment of any software system (Bittner

& Leimeister, 2013; Darch et al., 2009; Hoffmann et al., 2013; Tan, 1994). In this section we describe the types of shared understanding among a group of people, as well as some elements that contribute to achieve it.

2.6.1. Types of Shared Understanding

Glinz & Fricker (2015) identified two forms of shared understanding among participants of a requirements specification stage: *explicit* and *implicit*. Explicit shared understanding (ESU) is related to interpreting explicit specifications, such as requirements, design documents, and manuals, in the same way by all group members. Implicit shared understanding (ISU) refers to the common understanding of non-specified knowledge, assumptions, opinions, and values.

Research and practices in requirements engineering have contributed to achieve a good understanding of explicit shared understanding. However, implicit shared understanding is frequently neither clear nor reflected in current practice and research (Glinz & Fricker, 2015). For these reasons, the implicit shared understanding is more difficult to evaluate than the explicit one, in part by the subjective standpoint of each participant in the requirements specification process.

To exemplify both forms of shared understanding, we take the problem of a road construction site with one-lane traffic, controlled by two traffic lights described by Glinz and Fricker (2015). Assume that we need to develop a traffic light control system for managing alternating one-lane traffic, involving a team of stakeholders and developers. If the notion that a red light is a stop signal is shared by all team members, we have implicit shared understanding. If there exists an explicit requirement stating “The stop signal shall be represented by a red light”, we have explicit shared understanding, provided that all team members interpret this requirement in the same way. On the other hand, the team members might have misunderstandings (i.e., false implicit shared understanding or false explicit shared understanding) or they might omit relevant information (e.g., data not noticed by anybody), which is named “dark” information (Glinz & Fricker, 2015).

Although we might think to rely only on one form of shared understanding, in software development, we usually require implicit shared understanding and a set of explicit specifications to validate the artifact being analyzed and create agreements on it. Among the reasons for that are the complexity of the development process and the need of counting on explicit documentation of key concepts.

2.6.2. Practices to Achieve a Shared Understanding

Several practices have been developed to achieve a shared understanding between stakeholders and developers. Creating explicit artifacts contributes to achieving ESU and reduces the impact of false ISU. Artifacts such as documents and models, besides

of discussion, are the basis to identify and narrow the domain where shared understanding has to be achieved, and to obtain general understanding of important domain concepts (Glinz & Fricker, 2015; Schmid, 2002).

These explicit artifacts are required when feedback is performed to ensure shared understanding between sender(s) and recipients(s) of information (Cramton, 2001), and negotiation and prioritization is usually carried out to achieve explicit consensus on some concept or issue (Berander & Andrews, 2005; Grünbacher & Seyff, 2005).

Particularly, modeling (of domains, problems or solutions) makes the modeled concepts explicit and thus converts implicit shared understanding into explicit shared understanding (Gemino & Wand, 2005; Lauesen, 2002). Such models help infer and properly interpret non-modeled or only coarsely modeled concepts, and increase the probability of interpreting them correctly, thus contributing to the creation of proper implicit shared understanding. Modeling is a particular way of formalizing requirements or architecture (Glinz & Fricker, 2015).

There are some artifacts that can be useful to represent PDCPs, and thus to ease reaching a shared understanding about the product to be developed. These artifacts, usually known *conceptual frameworks* (or conceptual diagrams), provide semi-structured information about the product design, expressed as a combination of text and drawings, elucidating how the designers view a given problem and aim to develop a solution (Antunes et al., 2014). Conceptual frameworks are considered paramount artifacts for design knowledge representation (Goldschmidt, 1997). Several studies have shown the usefulness and benefits of these artifacts in the exploration of design problems and the corresponding solutions (Antunes et al., 2014; Comi & Eppler, 2011; Eppler, 2006; Ewenstein & Whyte, 2007).

Conceptual frameworks are considered as “artifacts of knowing” (Ewenstein & Whyte, 2007) because they help articulate and understand design ideas, and are material entities with which practitioners can interact as they generate knowledge individually or collectively. An example in which conceptual frameworks act as enablers for actors to negotiate their knowledge for the joint development of a shared understanding is illustrated in the research developed by Comi and Eppler (2011). In addition, in that work it is considered that during subsequent interactions among participants, physical contact with a conceptual framework can provide more information to renegotiate and refine common understandings.

Moreover, practices such as building and using a glossary (Glinz, 2014), using ontologies (Guarino et al., 2009), formalizing requirements or architecture in specifications (Glinz & Fricker, 2015), and prototyping (Wood & Silver, 1995) among others, contributes to achieve a shared understanding.

Specifically, an approved prototype is an artifact that explicitly represents a shared understanding of how a system to be shall look. On the other hand, a prototype also tests and fosters implicit shared understanding (Glinz & Fricker, 2015). (Britton & Jones, 1999) consider that a prototype facilitates communication between developers and clients or users. It also allows users to relate what they are shown to their own experience of tasks, and enables them to give meaningful feedback to the design ideas that are embodied in the prototype. These authors specify that when prototypes cannot be used or are not enough on their own, one of the main alternatives is currently to use one or more representations constructed using languages designed for specifying software.

Additionally, assessment practices are useful to validate explicit specifications; for instance, playing scenarios, model checking or systems prototyping. *Creating and playing scenarios*, i.e., examples of system usage, is a practice that helps assess and foster shared understanding about how a system will work in typical situations (Sutcliffe, 2010). *Model checking* is frequently used to determine whether some understanding of a specification actually holds (Clarke et al., 2018). *Prototyping or simulating systems* is useful for assessing and fostering shared understanding by comparing the expected and the actual system behavior (Seybold et al., 2005; Sutcliffe, 2010).

2.6.3. Relationship between Explicit Specifications and Explicit Shared Understanding

Explicit specifications and explicit shared understanding are strongly related to each other; however, they are not the same (Glinz & Fricker, 2015). The existence of explicit specifications, as well as of any other artifact, does not imply flawless explicit shared understanding. This is the reason why all specifications and other artifacts need to be validated. Only when an explicit specification has been validated, we can say that this specification constitutes explicit shared understanding.

2.7. Designing Effective Visual Modeling Languages

A model is a representation of reality, an abstraction of something relevant to the stakeholders, described using well-defined and unambiguous languages (Fisher et al., 2014). As discussed in the previous section, models have been recognized as useful tools to achieve a shared understanding among participants of the requirements engineering stage. However, counting with suitable models is far from easy. We next describe concepts about visual notations, as well as design principles to obtain an effective visual notation.

A visual notation (or visual language, graphical notation, diagramming notation) consists of a set of graphical symbols (visual vocabulary), a set of compositional rules (visual grammar) and definitions of the meaning of each symbol (visual semantics) (Moody,

2009). The visual vocabulary and visual grammar together form the visual (or concrete) syntax. Graphical symbols are used to symbolize abstract constructs (abstract syntax), typically modeled by means of a metamodel. The meanings of graphical symbols are defined by mapping them to the constructs they represent (Moody, 2009). The semantics (i.e., actual meaning) of abstract constructs can be formally (e.g. by means of a formal language) or informally (e.g. using natural language) defined.

During the last three decades, visual modeling languages have played a critical role in research and practice in software engineering. They are considered to convey information more concisely and precisely than ordinary language. However, designing them with *cognitive effectiveness* is far from simple. *Cognitive effectiveness* is defined as “the speed, ease, and accuracy with which a representation can be processed by the human mind” (Larkin & Simon, 1987).

Cognitive effectiveness determines the ability of visual notations to both communicate with business stakeholders and support design and problem solving by software engineers (Moody, 2009). In this section we describe the most important concepts about visual representations, and also a set of principles for designing cognitively effective visual notations.

2.7.1. The Importance of Visual Representations

Human information processing is highly sensitive to the exact form in which information is presented to the senses: apparently minor changes in visual appearance can have dramatic impacts on understanding and problem solving performance (Cheng, 2004; Larkin & Simon, 1987). Empirical studies in software engineering have confirmed that the visual form of notations significantly affects understanding, especially by novices (Irani et al., 2001; Irani & Ware, 2003; Masri et al., 2008). This suggests that decisions about visual representations are far from trivial and should be treated with care (Moody, 2009).

2.7.2. An Approach for the Development of Modeling Languages

The development of a Domain-Specific Modeling Language (DSML) is not a trivial task, given it often should feature a special visual notation (concrete syntax) that helps improve clearness and comprehensibility of models. (Frank, 2013) proposes a process that is intended to guide the development of DSML; it is constituted by the sequence of macro-processes shown in Figure 2.20.

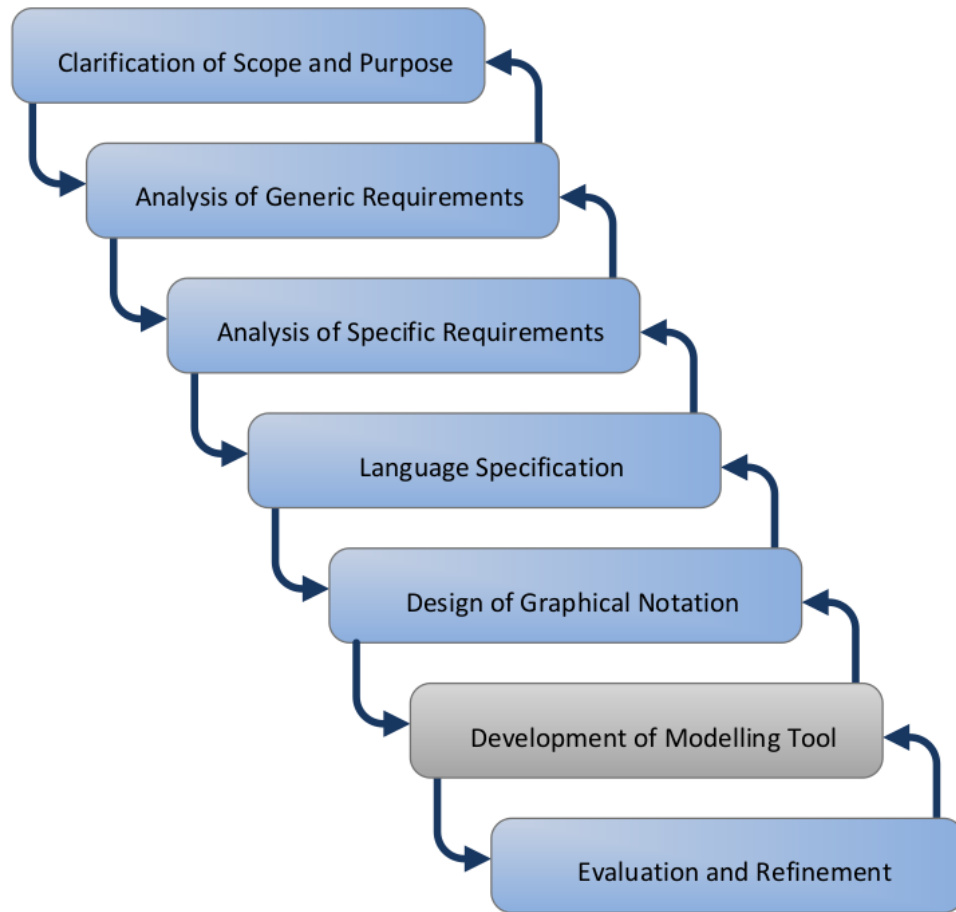


Figure 2.20. Macro-processes in the development of a DSML (Frank, 2013).

The macro-processes are briefly described next:

- *Clarification of Scope and Purpose.* This includes a high-level assessment of the benefits of the DSML with respect to certain classes of modeling tasks and corresponding users. Also, evaluating the costs and risks of introducing, using, and maintaining the DSML needs to be addressed.
- *Analysis of Generic Requirements.* Each requirement should be described and justified with respect to the purpose of the DSML. Moreover, each requirement should be characterized with respect to its relevance.
- *Analysis of Specific Requirements.* To analyze specific requirements, users, business analysts and language designers need to be supported; particularly, there should be a clear idea of what they may expect from the DSML. Designing preliminary diagrams could be helpful for this analysis.
- *Language Specification.* Its main goal is to specify and document a metamodel and corresponding constraints of the DSML.

- *Design of Graphical Notation.* For this activity it is suggested to use guidelines for the design of graphical symbols that represent modeling concepts, as well as notational elements requiring coping with the complexity created by large models.
- *Development of a Modeling Tool.* Although this macro-process could be left out because it requires focus on software engineering aspects (and probably to lose the primary focus, which is the development of the DSML), it was included in the sequence since it will often be necessary to make the DSML usable.
- *Evaluation and Refinement.* In order to ensure a certain quality level, it is mandatory to finally evaluate and possibly revise the DSML.

Frank (2013) recognizes that the sequence shown in Figure 2.20 is not the only way to structure the overall process (a proposition that could hardly be justified anyway), but this author claims that it is a proposal that makes sense. In such a structure, each macro-process could be characterized by further details (named micro-processes).

2.7.3. Design Principles for Cognitively Effective Visual Notations

Visual notations are usually defined without reference to theory or empirical evidence; when explanations about the elements of a notation are given, they tend to be based on common sense. Although some researchers have proposed some theories to evaluate visual notations (which refers to the last macro-process in Figure 2.20), none of them is considered as a standard.

In this work, we have considered the set of principles for the design of effective visual notations in Software Engineering proposed by Moody (2009). Such principles are named *the “physics” of notations*, and they can be used also to compare, and improve existing notations, as well as to develop new ones. For instance, it has been used to evaluate and improve several leading design notations, such as *ArchiMate* (The Open Group, 2019), *UML* (OMG, 2015), and *i** (Dalpiaz et al., 2016). Each principle is briefly described in Table 2.1.

Table 2.1. Principles for designing visual notations (Moody, 2009).

Design principle	Description	Comments about the principle
1. <i>Semiotic Clarity</i>	There should be a 1:1 correspondence between semantic constructs and graphical symbols	<p>The principle establishes when there is a 1:1 correspondence, or when one or more of the following anomalies occur:</p> <ul style="list-style-type: none"> ● <i>Symbol redundancy</i>: when multiple graphical symbols represent the same semantic construct. ● <i>Symbol overload</i>: when two constructs are

		<p>represented by the same graphical symbol.</p> <ul style="list-style-type: none"> • <i>Symbol excess</i>: when graphical symbols do not correspond to any semantic construct. • <i>Symbol deficit</i>: when there is a semantic construct that is not represented by any graphical symbol.
2. <i>Perceptual Discriminability</i>	Different symbols should be clearly distinguishable from each other	<p>This principle refers to the ease and accuracy with which graphical symbols can be differentiated from each other. For that, the following aspects of the symbols can be considered:</p> <ul style="list-style-type: none"> • <i>Visual distance</i>: It is measured by the number of visual variables on which they differ, and the size of these differences. • <i>Form</i>: It should be used as the primary visual variable to differentiate symbols. • <i>Redundant coding</i>: It establishes that visual distance between symbols can be increased by using multiple visual variables. • <i>Perceptual popout</i>: It determines that each graphical symbol should have a unique value on at least one visual variable to be detected without conscious effort. • <i>Textual differentiation</i>: This suggests that text is an effective way to distinguish between symbol instances, but not between symbol types.
3. <i>Semantic Transparency</i>	Use visual representations whose appearance suggests their meaning	This principle formalizes the informal notion of intuitiveness, which is often used to evaluate visual notations experimentally.
4. Complexity Management	Include explicit mechanisms to deal with complexity	This principle refers to the ability of a visual notation to represent information without overloading the human mind. It suggests that, to represent complex situations, the visual notations should provide <i>modularization</i> mechanisms (divide complex systems into subsystems) or <i>hierarchical structuring</i> (represent systems at different abstraction levels or details).
5. <i>Cognitive integration</i>	Include explicit mechanisms to support integration of information from various diagrams	This principle is tightly related to the fourth principle (i.e., complexity management) and it is based on the Perceptual Integrations and Conceptual Integration concepts. These two concepts are defined as follows:

		<ul style="list-style-type: none"> • <i>Perceptual integration</i>: If the notation counts on mechanisms to simplify the navigation and transitions among diagrams. • <i>Conceptual integration</i>: If the notation counts on mechanisms to help the reader assemble information from separate diagrams into a coherent mental representation of the system.
6. <i>Visual Expressiveness</i>	Use a complete range of capacities of visual variables	<p>This principle suggests using visual expressiveness to improve the discriminability (<i>principle 2, Perceptual Discriminability</i>).</p> <p><i>Visual expressiveness</i> is defined as the number of visual variables (i.e., shape, color, bright, size, texture, orientation) used in a notation.</p>
7. <i>Dual Coding</i>	Use text to complement graphics	This principle suggests using text together with the graphic elements of the language, to enrich such elements to help notation understandability.
8. <i>Graphic Economy</i>	The number of graphical symbols should be cognitively manageable	The principle suggests using around six categories of symbols.
9. <i>Cognitive Fit</i>	Use different visual dialects for different tasks and audiences	This principle states that different representations of information should be considered to different audiences or different tasks.

2.7.4. The notion of complexity in visual modeling

In this thesis work, “complexity” refers to diagrammatic complexity, which is measured by the number of elements (symbol instances or tokens) on a visual model. Complexity has a major effect on cognitive effectiveness, and thus, the amount of information that can be effectively conveyed by a single diagram should be limited by human perceptual and cognitive ability (Moody, 2009). Excessive complexity is one of the major barriers to stakeholders' understanding of software engineering models (Shanks & Darke, 1999).

The most common way of reducing the complexity of large systems is to divide them into smaller parts (or subsystems); activity that is usually known as modularization (Baldwin & Clark, 2000). This activity can take place in a top-down manner (e.g., when performing decomposition in Data Flow Diagrams) or bottom-up manner (e.g., when grouping components in UML packages). To avoid a cognitive overload on the system and process designers, the notations must provide mechanisms to divide large diagrams into perceptually and cognitively manageable “chunks”. Empirical studies show that

modularizing diagrams usually improves both the end-users' understanding and the capability of the designers to verify these specifications (Moody, 2009).

2.8. Model Consistency

Consistency is a general goal to be obtained while building the models (Muram et al., 2017), as it contributes to obtaining useful and non-contradictory models. It is defined as an internal property of an architectural model, which is intended to ensure that elements of that model do not contradict one another (Taylor et al., 2009).

2.8.1. A Classification of Model Consistency

Muram et al. (2017) in their systematic literature review (SLR), consider the following consistency types for software models:

- *Endogenous consistency*. This indicates consistencies within the same model, regarding the properties of itself.
- *Exogenous consistency*. It denotes consistencies between different models, and it can be refined into two subcategories: horizontal and vertical consistencies.
 - *Horizontal consistency* denotes consistencies among different kinds of models or a model against rules and constraints that are manually specified or derived from other artifacts.
 - *Vertical consistency* shows the consistency among models of the same type at different levels of abstraction. This also includes the consistency between a design model and a corresponding implementation.

On the other hand, existing studies distinguish between syntactic and semantic consistencies (Engels et al., 2001). Syntactical (or structural) consistency aims to ensure that a model conforms to a predefined syntax, specified by a certain meta-model or grammar. This ensures the well-formedness of the model.

Semantic consistency addresses the semantic compatibility of models, for instance, the same identifier that occurs in different models should refer to the same entity (i.e., have the same meaning) (Engels et al., 2001). Semantic consistency is stricter and often requires syntactical consistency beforehand; moreover, it depends on both the underlying semantics of the models being used and the development process (Muram et al., 2017).

2.8.2. Types of Consistency Checking Techniques

Another aspect to be considered in consistency checking is the used technique type. Muram et al. (2017) establish three types of techniques to carry out the consistency checking: *static*, *dynamic*, and *symbolic execution/simulation*. Next, we briefly explain each of them.

Static consistency checking techniques examine the models and/or their abstracted versions without running systems. The advantage of static checking is that it can be performed in early phases of software modeling and development, where no executable products are produced yet. However, static checking can be computationally expensive due to the cost of exhaustive analysis of large complex models.

In contrast, *dynamic consistency checking* aims to reveal inconsistencies while the system is running. Dynamic checking can achieve better computational performance as it only examines a small subset of system spaces (i.e., the actual execution traces). However, dynamic checking techniques require a running system and, in contrast to static checking, these techniques can not reveal all potential errors.

A third category of techniques named *symbolic execution/simulation consistency checking* uses symbolic execution or simulation approaches to validate the consistency of models. These approaches often consider a subset of input parameters and verify the response and consistency of the systems. As such, these techniques do not exactly belong to either of the aforementioned categories.

2.8.3. Main Consistency Checking Techniques Used in Existing Studies

Regardless of the type of techniques to check the model consistency (*static*, *dynamic*, or *symbolic execution/simulation*), four main techniques categories were defined, which have been adopted by existing studies (Spanoudakis & Zisman, 2001):

- *model checking*, i.e., using or combining with existing model checkers,
- *specialized algorithm*, i.e., utilizing algorithms designed for analyzing models to detect inconsistencies,
- *logical inference*, i.e., using formal inference techniques to derive inconsistencies, and
- *theorem proving*, i.e., applying reasoning using theorem provers.

Muram et al. (2017) found in their SLR, that a wide range of studies used specialized algorithms to identify and detect the inconsistencies of software models. According to these researchers, the main reason is that such approaches study different sets of inconsistency problems on particular domains and, therefore, often propose specific algorithms or heuristics for the particular contexts being studied (existing methods need to develop specialized algorithms for particular purposes).

2.8.4. Tool Support for Consistency Checking

An important aspect in consistency checking is to consider the tool support provided by the existing approaches. Given the human error-prone nature, ideally an automatic tool is required, not only to automatically check rules and constraints, but also to help users find and correct errors.

Muram et al. (2017) consider that more or better supporting tools would showcase a proof of feasibility, and attract the attention of practitioners to the corresponding techniques. The lack of any kind of supporting tool (e.g., research prototypes, demos) will hinder practitioners to use or at least explore proposed methods and techniques and, therefore, hinder the transfer of research results into industrial practice.

Existing tools and libraries have been frequently used to carry out consistency checking. Some of these model checkers are NuSMV (Clarke et al., 1996), SPIN (Holzmann, 1997), or GROOVE (Kastenberg & Rensink, 2006). However, during the last decade the use of these tools has decreased, compared with the use of other consistency checking methods like development of prototypes or tool-chains as illustrated in (Muram et al., 2017). These last two methods combine the implementation of various mechanisms that support model checking, such as model transformations and consistency checking algorithms.

2.9. Summary

This chapter described concepts related to the development of a system supporting a people-driven collaborative process, such as development strategies, requirements elicitation, and product scoping. Also, shared understanding between stakeholders and software engineers was discussed, which is a key aspect of this study. Additionally, concepts and principles about the design of effective visual modeling languages were described to understand how to assess visual notations. These concepts and principles can be used by notation designers to: design notations, compare notations, and evaluate and improve notations in a systematic manner. Finally, aspects related to model consistency were described to understand their importance in error identification, particularly when modeling is being performed.

Chapter 3. Related Work

Researchers from process engineering and computer-supported collaborative work communities have studied and proposed several notations and languages to specify interaction scenarios involved in people-driven collaborative processes. This chapter presents a review of modeling languages to specify interaction scenarios in PDCPs. Particularly, it introduces the most relevant ones for this modeling domain, and also presents an expert review of such modeling proposals. Next, we discuss the results of this review and the opportunities to improve the current modeling languages. Finally, threats to validity of this study are described.

3.1. Modeling Languages Description

In this section, we present the main characteristics of representative visual modeling languages and notations related to people-driven processes. Although, in this literature review several terminologies for people-driven processes were identified, they refer to similar characteristics of this kind of collaborative processes; particularly: 1) they lack a pre-established workflow that coordinates the individual actions of the participants, 2) the next action of a participant is unpredictable, and 3) the performed actions depend on both the context and the real-time decisions that participating people made.

We focused on two prominent research communities: the process engineering community and computer-supported collaborative work (CSCW) community, which have studied and proposed several notations and languages to specify several aspects of collaborative processes. Although there are positive results, most process modeling languages were not conceived to represent PDCPs (Allah Bukhsh et al., 2019; Dorn et al., 2014); therefore, we can expect that these languages show limitations to specify this type of processes.

3.1.1. Case Management Model and Notation

From a business process perspective, the OMG developed the Case Management Model and Notation (CMMN) (OMG, 2016). This is a visual modeling notation based on the Case Handling paradigm proposed by (Aalst et al., 2005), and adapted for modeling processes where the activities depend on real-time evolving circumstances. Here, a process instance is referred to as a Case, and the knowledge workers in charge of a particular Case actively decide on how the goal of that case is reached, and the role of a case handling system is assisting rather than guiding them in doing so.

A Case has two distinct phases, *the design-time phase* and *the run-time phase*. During the design-time phase, business analysts engage in modeling, which includes defining Tasks that are always part of predefined segments in the Case model, and “discretionary” Tasks that are available to the Case worker to be applied at his/her

discretion. In the run-time phase, Case workers execute the plan, particularly by performing Tasks as planned, while the plan may continuously evolve, due to the same or other Case workers being engaged in planning, i.e., adding discretionary Tasks to the plan of the Case instance in run-time (OMG, 2016).

CMMN does not have any notation to characterize assigned user roles and the interactions among them can only be established through notifications. For instance, Figure 3.1 shows a model representing a Case of an Electrical and Electronic Equipment exchange via a web platform (Routis et al., 2018). This model depicts the functionality that a collaborative platform provides to its users, highlighting the interaction between the various activities included in the Case.

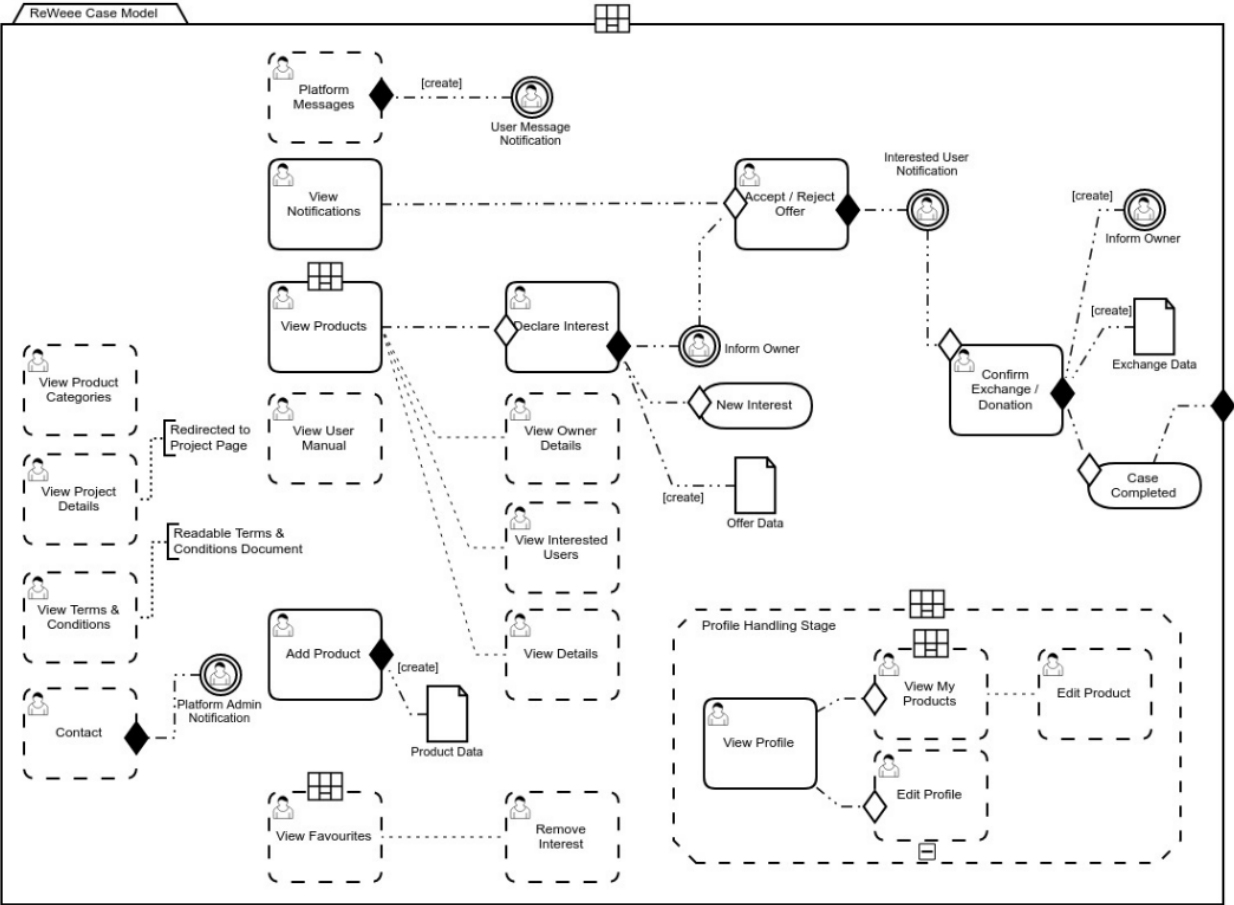


Figure 3.1. A CMMN model: Electrical and Electronic Equipment Exchange between users (Routis et al., 2018).

Indeed, the purpose of a CMMN model is to provide guidance to engineers about what can be done for successful process execution instead of defining design-time conditional flows. However, the same model specification is limited at design-time given the people-driven processes nature. Moreover, this notation does not provide mechanisms to represent the roles participating in the PDCP.

3.1.2. BPMN Plus

Taking advantage of the capabilities of the standard BPMN (OMG, 2014), (Allah Bukhsh et al., 2019) proposed an extension to such a modeling notation. This extension, named BPMN Plus, aims to be capable of modeling people-driven processes and proposes several interesting modeling concepts, for instance, *optional activities* (that can be skipped during the process execution considering the process context), *undo activities* (that need to be undone considering the particular process context), *events* (occurrences of real-world events related to a process), and *performers* (roles that should perform activities).

Compared to CMMN and BPMN, the constructs of BPMN Plus provide some benefits in the modeling of a business process in a flexible manner, for instance, the ability to model user roles, the possibility to specify business rules on the process model, and the capability to model collaborative activities.

Figure 3.2 shows an example of a generic admission process for applicants. The model includes the activities and also the interactions and dependencies (flow) among them. BPMN Plus represents, in a relatively simple manner, rules for particular user roles executing the process; for instance, the *admission administrator* role needs to decide about the completeness of the admission application before forwarding it to the *admission coordinator*. Also, collaborative sub-processes are represented in a straightforward way; for instance, ‘*discussion for decision*’ is a collaborative sub-process that shows the interaction among user activities.

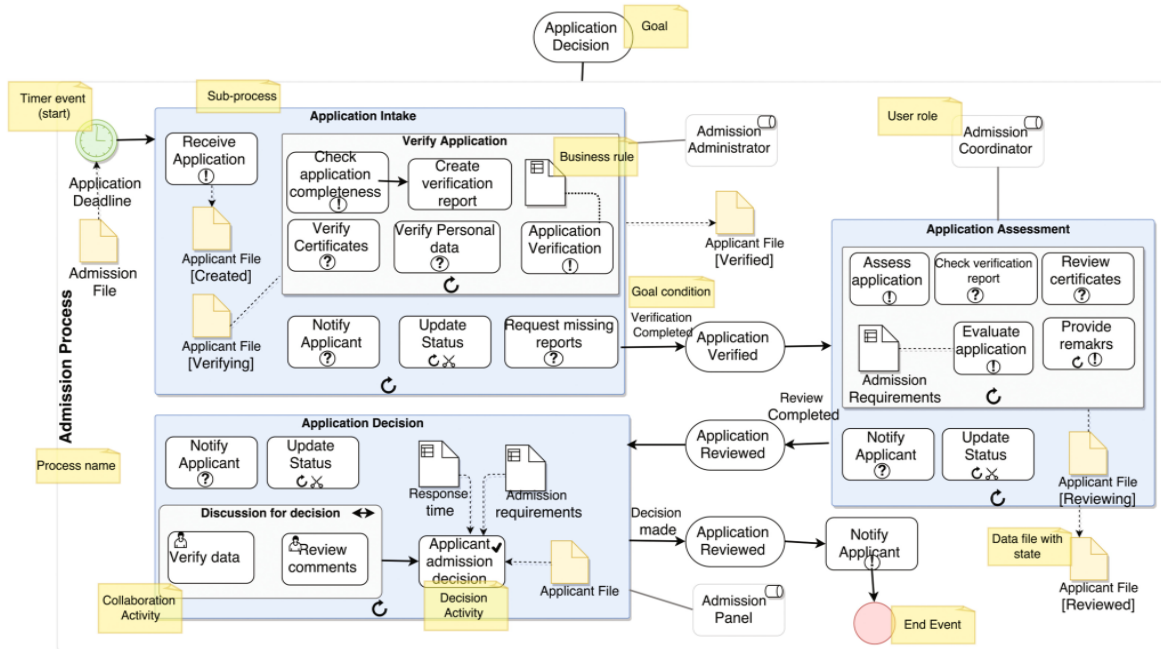


Figure 3.2. Model of an admission process (Allah Bukhsh et al., 2019).

Although this notation has shown to be useful and expressive for modeling specific aspects of people-driven collaborative processes, they have limitations to address some of the modeling challenges described previously. For instance, the notation is targeted to engineers and BPMN experts, consequently hindering the shared understanding between engineers and stakeholders.

3.1.3. BPMN4SBP

Similar to the previous notation, Hassen et al. (2019) proposed an extension to BPMN, named BPMN4SBP. This extension, which is a Sensitive Business Process Specification based on core ontologies, integrates and implements the six modeling dimensions of a generic Business Process Meta-model for Knowledge Identification (BPM4KI) (Ben Hassen et al., 2017), which were conceptualized for describing a Sensitive Business Process: the *functional*, *organizational*, *behavioral*, *informational*, *intentional* and *knowledge* dimensions. Thus, it allows addressing the dynamic, interaction and knowledge aspects of such processes. Due to their characteristics, we will refer to Sensitive Business Processes as people-driven processes.

The graphical notation used by BPMN4SBP contains elements for representing specific aspects of a PDCP, for instance, (individual) *human* and *collective roles*, *tacit* and *explicit knowledge*, and distinct kinds of *knowledge flow*. An example of a model is depicted in Figure 3.2. It shows the process of a neuro-motor evaluation of a child with cerebral palsy (Ben Hassen et al., 2019).

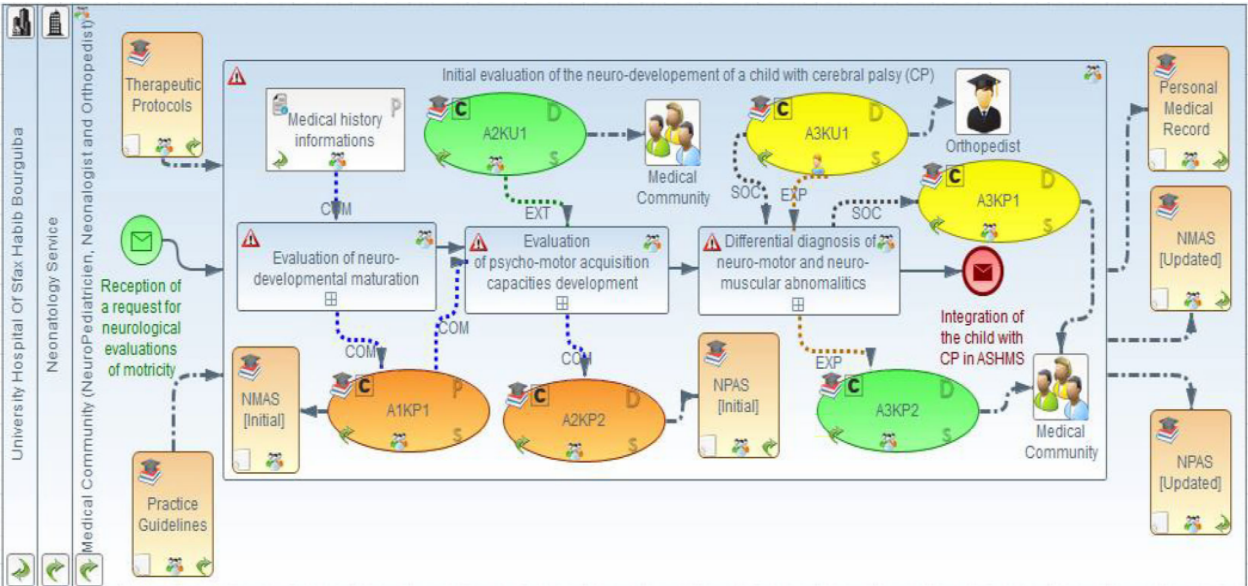


Figure 3.3. Model of the initial neuro-motor evaluation process for a child with cerebral palsy (Ben Hassen et al., 2019).

This process strongly relies on the tacit knowledge embedded in the stakeholders' minds (i.e., the orthopedist and other members of the medical community). This is conveyed to other participants through distinct kinds of flows (e.g., SOC that represents socialization, or COM that represents communication).

As evidenced, BPMN4SBP has several useful elements to model people-driven processes, together with a strong expressivity. However, such as BPMN Plus, it has limitations to address several of the modeling challenges described previously.

3.1.4. Little-JIL and hADL

Some other visual notations have emerged from the CSCW research area. They consider the specification of interactions among participants as a key aspect in a collaborative process. Particularly, Dorn et al. (2014) proposed the joint usage of two human-centric specification languages to model interaction-intensive processes. These languages are Little-JIL (Cass et al., 2000), that is a process-centric language, and hADL (human Architecture Description Language) (Dorn & Taylor, 2012), that is a structure-centric human interaction language.

On the one hand, Little-JIL depicts processes as hierarchies and sequences of steps and each step may itself be an entire step hierarchy. It also incorporates step execution sequencing specifications (e.g., sequential and parallel activities) and exceptions handling (exceptions may be thrown by an execution agent, which may be a human or a software entity, and steps can contain one or more exception handlers).

On the other hand, hADL describes how humans interact among them to achieve a common goal, for instance, to discuss and then to jointly decide upon resource usage and next steps. This language distinguishes between elements as *human components* and *collaborator connectors*. Thus, it emphasizes the difference between the primary collaborating users (e.g., a decision maker) and non-essential, replaceable users that coordinate the collaboration (e.g. a discussion moderator). Other elements, as *collaboration objects*, abstract from concrete interaction tools and capture the semantic differences in subtypes (e.g., *messages*, *streams*, or *shared artifacts*).

Figures 3.4 and 3.5 illustrate a healthcare emergency process using these notations. Particularly, Figure 3.4 shows a Little-JIL model consisting of the hierarchical/sequential steps of the emergency process. For instance, *PutPatientInBed* is a step executed if the *AssignBedScope* step was performed previously and the *ExceptionHandler* does not report a failure. Figure 3.5 presents a hADL model, focused on the coordination among *physicians*, *supervisor*, *nurses* and process steps agents, when to switch assessment strategies and also when they volunteer for role substitution.

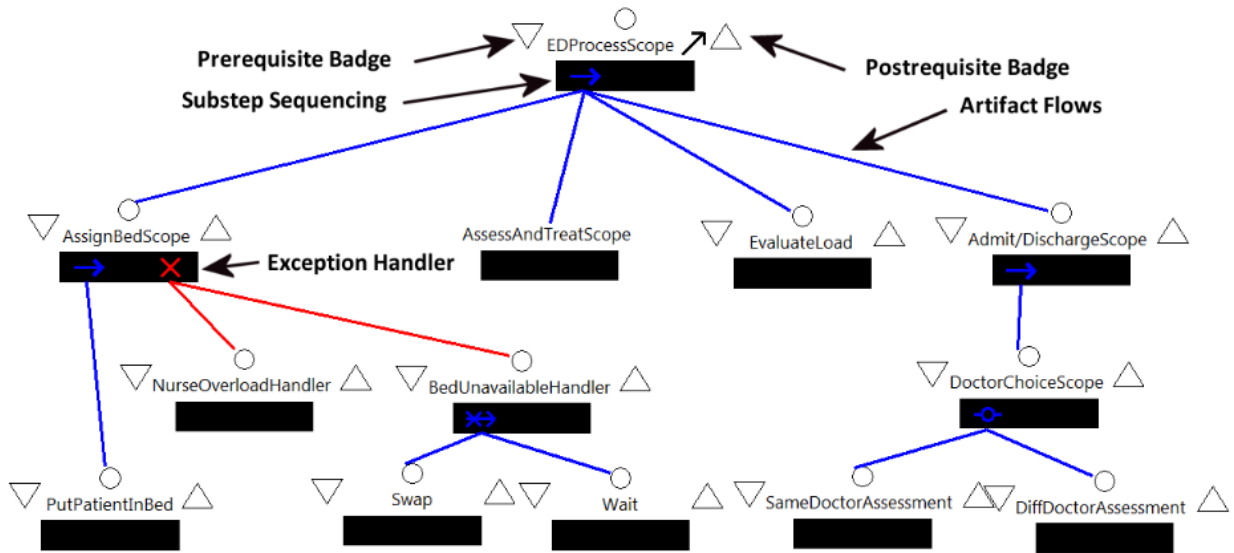


Figure 3.4. Model representing hierarchical steps specified for an emergency process using Little-JIL (Cass et al., 2000).

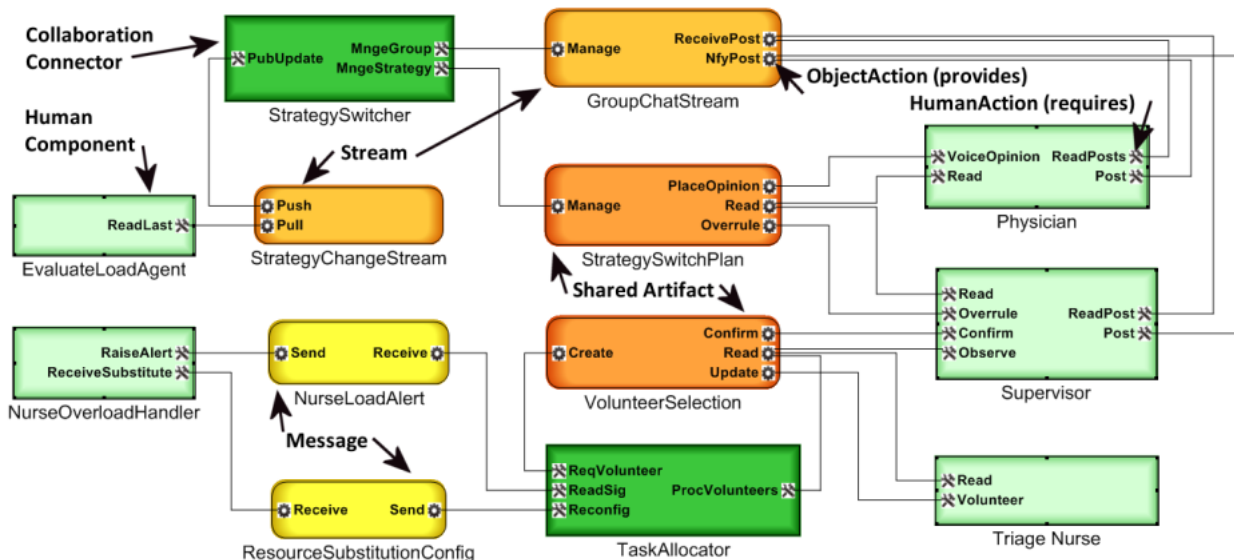


Figure 3.5. Model representing interaction among participants in an emergency process using hADL (Dorn & Taylor, 2012).

Although these languages have important advantages over other languages (BPMN among them), their joint usage increases the complexity of the modeling process. Therefore, it decreases the feasibility to include stakeholders in the process specification, analysis and validation. Moreover, each language proposes a single visual dialect that is designed for technical people. The joint usage of these languages requires software tools that perform the models' integration. This is why a complete model consistency check is performed only partially.

3.1.5. Mobile Collaboration Modeling

In the same field, other approaches that specify actors and interactions among them, have shown a higher suitability to represent the interaction scenarios in people-driven collaborative processes. For instance, the Mobile Collaboration Modeling (MCM) notation (Herskovic et al., 2019) represents these scenarios through a directed graph, in which the nodes represent the roles (i.e., actors participating in a people-driven mobile collaborative process) and the arcs represent the interactions among them.

The nodes of the graphs may be active users (i.e., people) or intermediaries (e.g., a server synchronizing the results of human activities) participating in the interaction scenario. Apart from that, each arc is labeled with a notation that describes the interaction needs among the roles. It represents the possible scenarios in which interaction would take place.

These scenarios correspond to a classification scheme with two dimensions: *simultaneity* and *reachability*. The first one is related to the time period in which the actors performing a role are potentially available to interact among them. The second dimension is related to the availability of an interaction channel (physical or virtual) between the roles involved in a potential interaction. Figure 3.6 presents a MCM model representing the interactions between roles involved in an urban search-and-rescue process. In MCM, the interactions are labeled with squares specifying the simultaneity and reachability between actors (e.g., the interaction between the roles *Commander* and *Operations* is simultaneous and reachable).

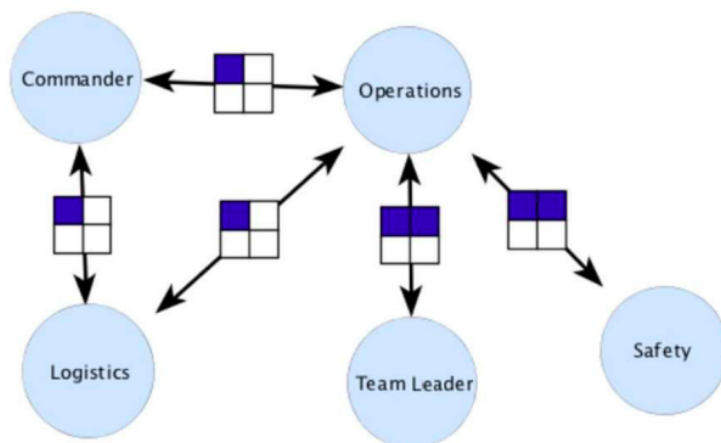


Figure 3.6. MCM model for an interaction scenario involved in an urban search-and-rescue (USAR) process (Herskovic et al., 2019).

This notation was conceived to specify and validate PDCPs, and to be used by designers and developers, more than by stakeholders. It includes limited interaction information and does not consider mechanisms for managing the complexity or size of

the PDCP, which is one of the design challenges that should be addressed by these modeling notations.

3.1.6. IoT Modeling Notation

Monares et al. (2014) defined a modeling notation to represent interactions in human-centric wireless sensor networks. This notation is based on the proposal of Herskovic et al. (2009), which represents scenarios involved in people-driven mobile collaborative processes through graphs. Similar to MCM, in IoT modeling the nodes represent the roles (i.e., the actors participating in the process) and the arcs represent the interactions (or relationships) among them.

This notation allows designers to model complex interactions between network nodes, which can be: human-based sensors, mules (particular mobile units), witness units, regular sensors, or actuators. The arcs are stereotyped to indicate the level of simultaneity (i.e., simultaneous, overlapped, or disjoint) that the presence of two nodes has in the modeled scenario. The intermittence of the arcs (i.e., if they are permanent or intermittent arcs) is determined by the type of communication support available to perform the interactions between the involved nodes.

Figure 3.7 illustrates a model representing an IoT system, used to try mitigating felony acts in a certain neighborhood. As specified, each node interacts with another node with a specific simultaneity level (e.g., every *neighbor* can interact, in a simultaneous manner, with the *911 operator*). In that system, the neighbors can use the video-camera and GPS embedded in their smartphones (i.e., regular sensors) to report incidents or georeference information about incidents.

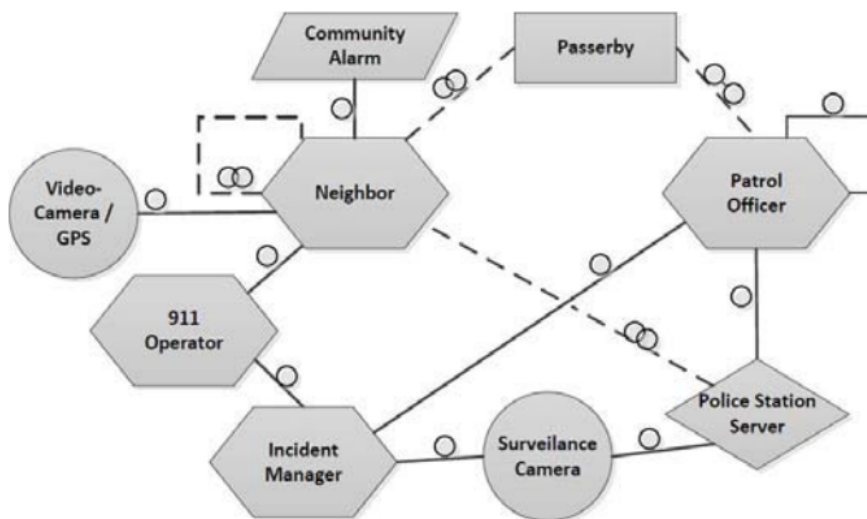


Figure 3.7. IoT Interaction model for a personal security system (Monares et al., 2014).

This notation is also targeted at developers, and it was designed for specifying human-centric wireless sensor networks; the interaction information specified in these models is limited. Similar to MCM, it does not consider mechanisms for managing the complexity or size of the collaborative process.

3.1.7. Collaboration Graphs

Hawryszkiewicz (Hawryszkiewicz, 2005, 2009) proposes an approach, based on collaboration graphs, that extends social network diagrams. The notation was adapted to specify integrated business activities. In order to manage the models' complexity and size, this approach uses a combination of business activities, interaction graphs, and knowledge requirements as the three basic constructs to specify the interaction models.

Figure 3.8 shows a collaboration graph that represents the responsibilities and interactions on process outsourcing. The roles are shown as dots, whereas the participants are shown as faces; for example, 'a1' is the leader of the provider team. The collaboration within activities is shown using the circles, and the role responsibilities are represented by the dotted boxes linked to the roles. Finally, the interactions between the roles are shown with dotted boxes, linked by dotted lines to the corresponding interaction.

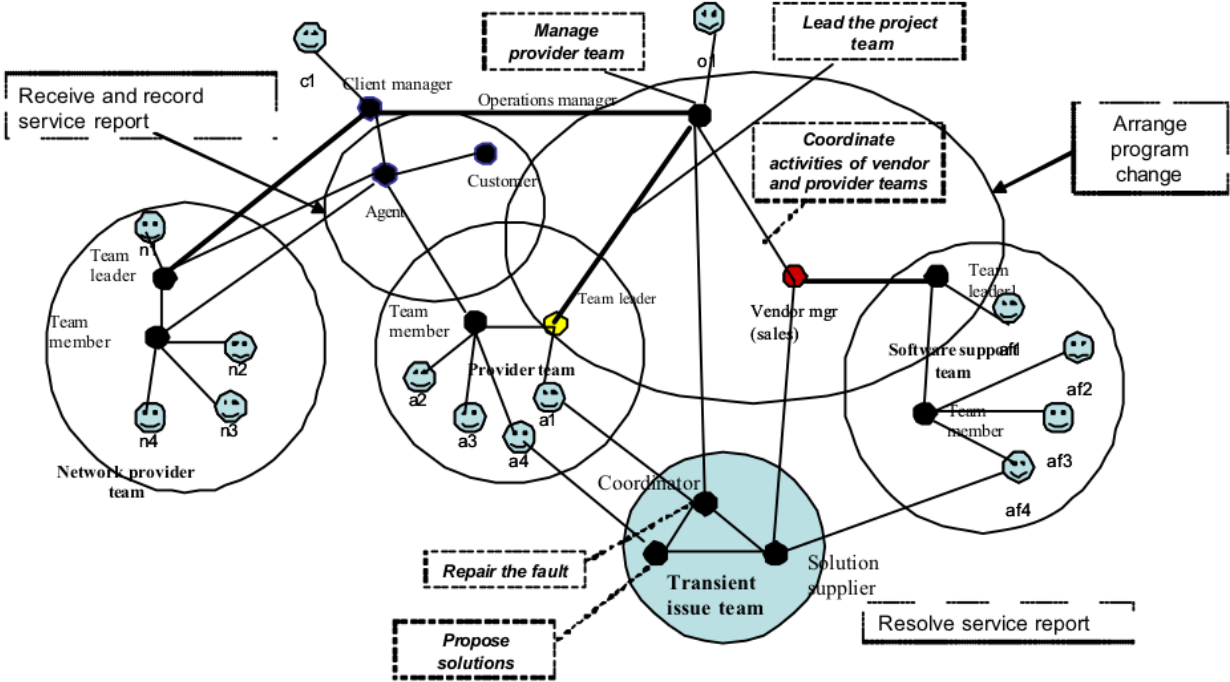


Figure 3.8. Model of an outsourcing process using collaboration graphs (Hawryszkiewicz, 2009).

Although this proposal provides several valuable insights to model interactions, collaboration graphs are not easy to understand by stakeholders, therefore, they are not very suitable to build shared understandings between developers and stakeholders. Moreover, the modeling concepts considered in this proposal correspond to integrated enterprises and, similar to the two previous proposals, the information specified through these interactions' representations is limited. This reduces the capability of developers to derive software requirements from them.

3.1.8. Computer-mediated Interaction Modeling Notation

Computer-mediated Interaction Modeling Notation (CIMoN) visually represents the roles played by the participants in dynamic work scenarios involved in PDCPs (Canché et al., 2019; Canché & Ochoa, 2018). This notation defines interaction graphs that are the result of a design activity among developers and stakeholders.

The analysis of these graphs allows determining and agreeing the set of interaction services that should be embedded in a mobile application that supports a particular PDCP. This notation allows specifying essential aspects, e.g., the interaction type between roles (synchronous, asynchronous or both of them), messages type, and the availability of the participants to interact with others.

Figure 3.9 shows a CIMoN model for an urban emergency response process. The nodes correspond to each participating role, and the edges represent the type of link and interaction requirements between them. For instance, team captains are human actors that need to interact synchronously among themselves. All computer-mediated interactions among most human actors are synchronous and conducted through an intermediary node (the communication server). The roles interaction (RI) graph also shows that there are only three mandatory roles, i.e., incident commander, team captain and communication server. The graph establishes the minimum number of actors required to perform the process. The circle in the upper left corner of the node indicates such a cardinality; no circle means an optional role.

Although CIMoN was conceived to address the limitations of the previous notations, it did not reach such a goal. For instance, although the model seems to be understandable for stakeholders, its representation still needs some improvements to reach shared understandings between developers and stakeholders (Canché et al., 2019). Moreover, the notation does not consider mechanisms for managing the complexity and size of an interaction model representation.

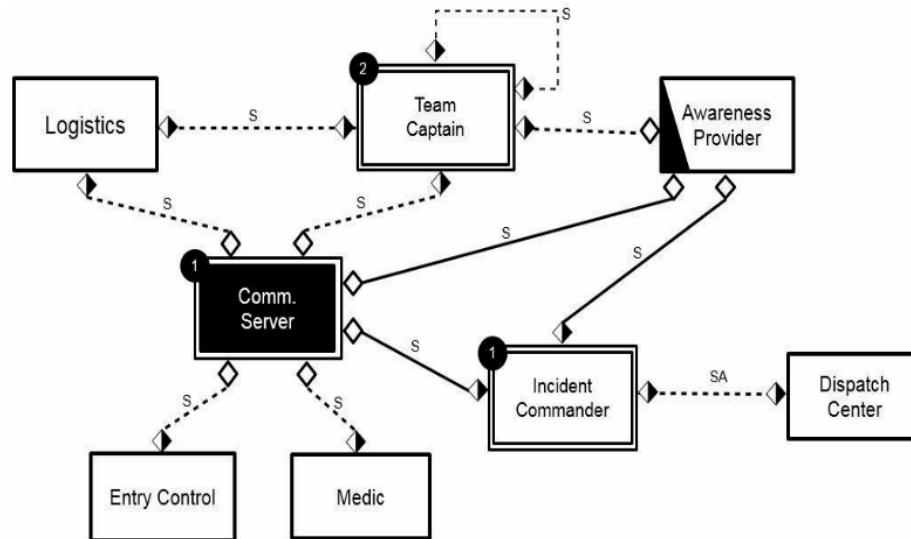


Figure 3.9. An interaction graph for an urban emergency response process using CIMoN (Canché & Ochoa, 2018).

The following section presents a comparative review, based on the opinion of the experts in these notations, about the accomplishment of these languages with respect to the challenges presented in Section 1.3.

3.2. Expert Review of Model Representations

In most cases, the interviewed experts were the authors of the modeling languages and notations. An instrument and its rubric were developed and sent to these experts to obtain their assessment. The reason to involve researchers at this stage was mainly by counting on a different and expert perspective of our assumptions about the challenges modeling addressed by the existing works. Table 3.1 presents a summary of the comparative assessment, based on the answers of these researchers. The labels in the table mean: 'Yes' - the notation addresses that challenge; 'No' - there is no evidence that the notation addresses the challenge; and 'Limited' - the challenge is partially addressed by the notation or language.

The first five challenges should be addressed by the modeling language, and the last two should be dealt with software tools that support the modeling process. As Table 3.1 illustrates, no language fulfills all the aspects, however certain positive assessment was considered by the researchers for some languages. As expected, regarding the first capability, all notations were able to represent roles and interactions among them in their models, with exception of CMMN that addressed it partially.

Concerning capability of the models to be understandable for stakeholders, most proposals provide it at least in a limited form, except CIMoN that address it properly. The authors of BPMN4SBP and BPMN Plus considered that this aspect is addressed by

those notations when the stakeholders are already experienced in the use of BPMN. As a consequence of the results shown for the second evaluation item, the reviewers considered all notations limited to help build a shared understanding between developers and stakeholders, about the structure of the interaction scenarios to be supported with the mobile application (or involved in the PDCP).

Table 3.1. Coverage of modeling challenges by the languages and notations.

Notation \ Aspect	(1) Capability to specify and characterize roles and interactions between them	(2) Capability to be understandable for stakeholders	(3) Capability to achieve a shared understanding of the specifications	(4) Capability to represent the specifications at different abstraction levels	(5) Capability to manage the size and complexity of the specifications	(6) Capability for automatic model consistency checking	(7) Capability for automatic derivation of software requirements
CMMN (OMG, 2016)	Limited	Limited	Limited	Yes	Yes	Limited	Limited
BPMN Plus (Allah Bukhsh et al., 2019)	Yes	Limited	Limited	Yes	No	Limited	Limited
BPMN4SBP (Ben Hassen et al., 2019)	Yes	Limited	Limited	Yes	Yes	Limited	Limited
Little-JIL & hADL (Dorn et al., 2014)	Yes	Limited	Limited	Yes	Limited	Limited	Limited
MCM (Herskovic et al., 2019)	Yes	Limited	Limited	No	No	Yes	Yes
IoT Modeling (Monares et al., 2014)	Yes	Limited	Limited	No	No	Limited	Limited
Collaboration Graphs (Hawryszkiewicz, 2005, 2009)	Yes	Limited	Limited	Yes	Yes	No	No
CIMoN (Canché et al., 2019; Canché & Ochoa, 2018)	Yes	Yes	Limited	No	No	Yes	Yes

Regarding the fourth aspect (capability to represent the specification at different abstraction levels), 5 out of 8 modeling notations addressed it. In the case of CMMN, BPMN Plus and BPMN4SBP, this capability is inherited from the source language in which they are based on.

Barely different from the previous aspect, the capability to manage the size and complexity of these specifications is fulfilled by only three languages/notations: CMMN, BPMN4SBP and Collaboration Graphs. The remaining ones do not provide this capability (such as BPMN Plus, MCM, IoT Modeling and CIMoN) or address it in a partial manner (such as Little-JIL & hADL).

Regarding the last two modeling challenges, i.e., the capability to perform automatic model consistency checking, and the capability to conduct automatic derivation of interaction services, the notations address them at most in a limited form, except MCM and CIMoN that cover them properly.

Summarizing, these results show an opportunity to improve the existing modeling languages, or define new ones, to improve the coverage of the stated modeling challenges. This is also evidenced in the software tools that support the activity of modeling by using these languages and notations.

3.3. Discussion

Several modeling languages, notations, and also extensions to the existing ones, have been proposed to represent the communication aspects in PDCPs. These representations make explicit the roles of the actors participating in the process, and also make evident some interaction requirements of these roles to collaborate among them. However, these modeling languages and notations still have limitations that jeopardize the shared understanding of the obtained representations. Consequently, it limits the capability of developers and stakeholders to analyze, validate and refine the interaction scenarios, and finally, to derive interaction requirements from such scenarios. These facts open an opportunity to improve the existing modeling interaction representations, or define new ones, to improve the coverage of the stated challenges.

3.4. Threats to Validity

The language/notation analysis we performed poses the following threats to validity:

Internal validity: Some factors may influence this comparative analysis. Among them, we can mention the following: participants' selection, given that such a process was performed with a direct invitation via email to researchers to assess their own work, which could generate a bias in their opinion. Also, some experts may be unwilling or unmotivated to participate in this analysis.

External validity. We are aware that the participation of the thesis author as coordinator of the study may have biased the obtained results. However, the participation of such a person comes down to explaining the evaluation rubric and clarifying its meaning if required.

This work recognizes that the number of participants (one person by modeling proposal) and the received subjective opinions, poses a threat to the generalization of the results. However, we consider that the participation of an author of a related work, or an expert related to a specific modeling language/notation, is a valuable contribution for this analysis.

Reliability. In order to decrease the potential impact of this threat to validity, the rubric was designed by the thesis author and reviewed and refined by two additional researchers. Also, the collected data were analyzed and discussed by the author and those two researchers, spending enough time with each assessed item, and giving sufficient concern to analysis that could contradict the findings.

3.5. Summary

This chapter presented the main characteristics of modeling languages and notations that can be used by designers of collaborative applications to represent the interaction scenarios among the participants of a people-driven collaborative process. Specifying these scenarios is a prerequisite to determine the user roles (user profiles) and the interaction capabilities (interaction services) to be considered in the mobile application to be developed.

An expert review of the visual modeling languages and notations was performed, which considered a set of capabilities that are challenging for language designers. This set of capabilities was drawn from the analysis of the literature in the study domain, and it also represents a contribution of our work.

The results show many opportunities to improve the current languages or define new ones. They also can be used to support practitioners and researchers of mobile collaborative systems during the selection of appropriate languages to represent their interaction models, depending on their current needs (or mandatory capabilities for their representations).

Chapter 4. The Computer-Mediated Interactions Modeling Language

The Computer-Mediated Interactions Modeling Language (CIMoL) was designed to model interactions among roles in scenarios involved in people-driven (or *knowledge intensive / unstructured*) collaborative processes (PDCP). CIMoL considers the modeling challenges reported in the literature to design systems and uses the principles of visual languages design proposed by Moody (2009), described in Chapter 2. The following sections present the main structure, nomenclature and design decisions of the language, and also the way to use it to create the interaction models involved in PDCPs.

4.1. Language Foundations

The *process* is the major element to be modeled using CIMoL. This element represents a PDCP. It is composed of *phases*, which are sequential and of which there should be at least one per process. Each phase is constituted by *work ambits*, which are interaction scenarios that can be present in parallel, and each of them can be involved in more than one phase. Each work ambit is represented by an interaction graph where the nodes represent the actors (i.e., the roles participating in the collaborative process) and the arcs represent the potential interaction requirements among them.

In order to conceive these graphs, the following assumptions should be considered:

- The process participants (represented by the nodes) are both autonomous and multitask units that auto-regulate their activities according to their own criteria, but considering the general business process being performed and also the current state from their own local and process contexts.
- The autonomous units manage a list of pending activities that are prioritized according to criteria defined by each participant (e.g., its urgency level, if the activity is critical, etc.). The priorities of activities change over time, due to external factors and also by the local actions from each unit.
- Since the work context is infinite (McCarthy, 1993), some awareness services (that provide interaction supporting information) can be conveyed to the autonomous units, and consequently they can interpret such information. Based on that, each unit decides its next action.
- The autonomous units dynamically decide, according to their current context, the workflow to be conducted and the temporality of their actions.
- The modeling notation only represents the potential computer-mediated interactions to be supported.

4.2. Language Metamodel

A metamodel was created to describe the abstract syntax of CIMoL (Figure 4.1). It represents the main concepts to be used in CIMoL models, and the relationships among them. A UML class diagram was used to specify the metamodel.

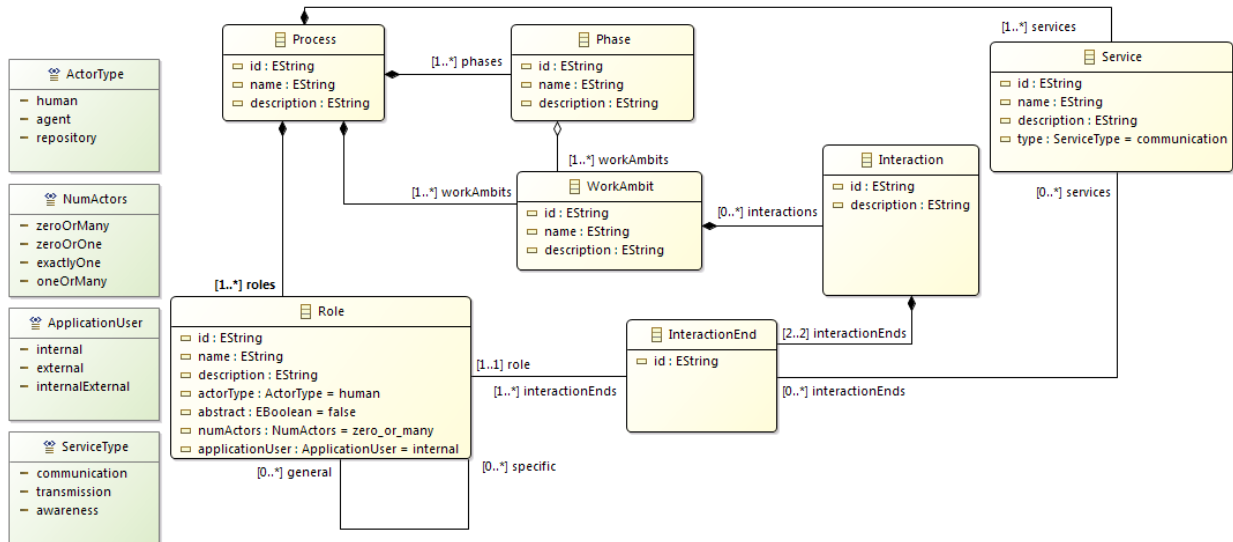


Figure 4.1. UML representation of the CIMoL metamodel.

As mentioned in the previous section, the major concept in the metamodel is *process*, which represents a people-driven collaborative process, i.e., the main work environment of the remaining concepts. Such concepts and their relationships are described as follows:

- Each process contains one or more roles (*Role* concept). Each role represents a type of actor *actorType* participating in the process. The process also includes one or more services (*Service* concept) that represent the software services required by the roles while performing the PDCP.
- Each process is composed of one or more phases (*Phase* concept), which can be considered as macro-activities of PDCP.
- Each phase is composed of one or more work ambits (*WorkAmbit* concept), which embed the interaction scenarios to be modeled. The work ambits can be present in a PDCP in parallel, and they can belong to more than one phase.
- A work ambit is composed of one or more interactions (*Interaction* concept) between roles (*Role* concept). Each interaction contains two ends (*InteractionEnd* concept), and each end can be associated with specific software services (*Service* concept) that correspond to a particular role (*Role* concept).

- Finally, a role (specific relationship) can be associated to zero or more roles (general relationship), which correspond to a relationship comparable to an is-a relationship from UML.

Structurally, the metamodel states that a process is composed by roles, phases, work ambits and services, and transitively, by the interactions of its work ambits, where each interaction has two interaction ends. However, as a model conforming to the metamodel can define more than one process (i.e., the model can have more than one instance of Process), we also specify invariants to state that the elements composing one process cannot be part of any other process (i.e., in a model, processes do not share their composing elements). The invariants constraining models conforming to the metamodel are the following:


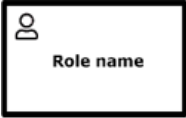




- The phases of a given process X aggregate only work ambits composing the process X.
- The roles of a given process X can only participate in interactions (through interaction-ends) that are composed in the work ambits composing the process X.
- The interaction ends of a process X can only use services composing the process X.
- Every pair of roles participating in the general-specific association between roles belong to the same process.
- The general-specific association between roles is acyclic, that is, a role cannot be more general (or more specific) to itself, neither directly nor indirectly.
- Every interaction must have at least one of its interaction-ends with at least one associated service.

4.3. CIMoL Common Constructions

CIMoL includes two views (or dialects): stakeholder view and developer view. The first one is targeted to stakeholders, whereas the second one was designed to be used mainly by developers. Regardless of the dialect, each model is based on the graph concept, in which the nodes represent the roles performed by the actors (i.e, the autonomous units) participating in the PDCP, and the links indicate the potential interactions between them.


This notation allows stakeholders and software engineers to represent each relationship between two nodes, as a point-to-point relationship. This reduces the complexity of designing the interaction scenarios to be supported by collaborative systems. Table 4.1 illustrates the visual representation of the node types considered in both dialects.


Table 4.1. Node types considered in the two dialects of the language.

Node type (Stakeholder view)	Node type (Developer view)	Meaning
 Role name	 Role name	<i>Human actor:</i> Person that uses the system to play a particular role during the PDCP. These nodes are able to interpret their own work contexts and take actions accordingly. The availability of these units to collaborate with others is defined by themselves, depending on their work context at the moment that a collaboration request is received.
 Role name	 Role name	<i>Autonomous agent:</i> Autonomous software component that behaves according to a preset list of actions. These actions can be context-aware or context-independent. It is assumed that these units are always available to collaborate when required.
 Role name	 Role name	<i>Repository:</i> Passive software component (e.g., a data repository) that only stores data and produces answers to requests that were triggered by human actors or autonomous agents. Similar to the previous case, it is assumed that these units are always available to collaborate and also enable others to do so.

Additionally, for both dialects, a *is-a* relationship is used to represent that actors performing a specific role B are also considered as being of a more general role A; i.e., role B extends from role A. Also, an *interaction* relationship between two nodes is used to represent a physical or virtual channel of communication that is available at the time they decide to collaborate. No interaction relationship is represented if there are no communication requirements between two roles. These two types of relationships are described in Table 4.2.

Table 4.2. Types of relationships between nodes considered in the language notation.

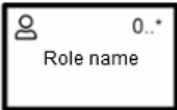
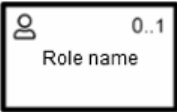
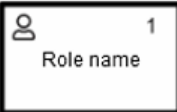
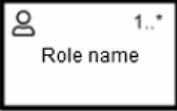
Relationship type	Meaning
	There is an interaction relationship between two nodes. This means that one participant requires communication with the other, and possibly, viceversa. A relationship on the same node (cycle) represents an interaction among actors playing the same role.

	<p>There is an is-a relationship similar to that in UML. The node without an arrow (the “specific” role) can be regarded as the role with a white arrow (“general” role). Thus, actors of the specific role also play the general role. As a consequence, the interactions and services available for the general role are also available (i.e., inherited) by the specific role. Both roles in the relationship must be of the same node type. Moreover, cycles are not allowed in the is-a hierarchy.</p>
---	---

4.4. CIMoL Notation for the Developer View

For the developer view, the nodes’ representation contains additional information that helps engineers analyze the interaction graphs. In this view, both optional and mandatory roles are visually labeled, as well as the number of actors (participants per role) required to perform the process. Table 4.3 shows this representation. Even though the table shows the visual information for a human-actor node type, an analogous representation can be used for the other node types using the corresponding icons shown in Table 4.1.

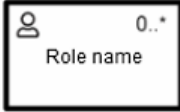
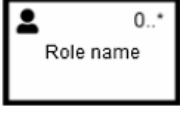
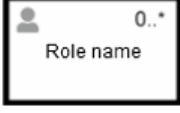
Table 4.3. Representation of the cardinality per role.

Symbol in node	Meaning
	<p><i>Zero or many.</i> The number of actors (role instances) to conduct the process can be zero or more. If the minimum cardinality for the role is zero, it means that the role is optional.</p>
	<p><i>Zero or one.</i> The number of actors to conduct the process can be zero or one. In this case, it also applies the optionality of the role indicated before.</p>
	<p><i>Exactly one.</i> The number of actors must be exactly one, and therefore, the role is mandatory.</p>
	<p><i>One or many.</i> The number of actors must be at least one (i.e., the role is mandatory).</p>

Additionally, the *user type* is visually labeled according to the application use for each node, which aims to specify whether an actor playing a role is going to use the application to be developed or not. Table 4.4 illustrates this representation.

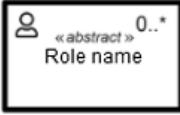
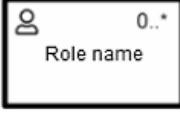
A white-figure role represents an internal user, a black-figure means the actor playing the role corresponds to an external user, and a gray-figure role indicates two things: that actors playing the role will use both the application to be developed and an external application, and/or that there are actors that will use the application and also there are other actors that will not. Although in the table the visual information is represented with a human-actor node type, and with zero-or-many instances, the symbols change according to the information that needs to be represented.

Table 4.4. Representation of user type according to the application use for each node.

Symbol in node	Meaning
	<p><i>Internal user.</i> Actors playing this role will use the (under development) collaborative application to interact with internal and external users.</p>
	<p><i>External user.</i> Actors playing this role will not use the (under development) collaborative application. They will use an external application (e.g., a commercial one) to interact with the internal users.</p>
	<p><i>Internal/External user:</i> These actors will use the (under development) collaborative application, and eventually an external application also to interact with other user types. Specifically, either the same actor will use both kinds of applications, or some actors will use the application to be developed while some other actors will not.</p>

Additionally, in is-a relationships, the *abstract* roles are visually labeled. By default, a role is considered *concrete*. When a role is specified as abstract in an is-a relationship, it means that this role will not be instantiable, that is, it will not have actors performing the role as it is, and instead, actors must be of any of the “specific” roles to carry out its activities. Table 4.5 shows this representation. Similar to previous cases, the other symbols shown in the roles’ representations depend on the information that the designer wants to communicate to the reader.

Table 4.5. Representation of abstract and concrete roles.

Symbol in node	Meaning
	<p><i>Abstract role.</i> The role does not have its own direct instances, that is, actors playing this role must be existing actors from other roles connected by an is-a relationship.</p>
	<p><i>Concrete role.</i> The role has its own instances. By default, each role is a concrete role.</p>

4.5. Interaction Services Specification

Concerning the stakeholder and the developer views, the *communication*, *interaction awareness* and *transmission services* must be defined for each node to interact with other nodes. Developers can define the set of services that are available in the process, as indicated in the metamodel by the metaclass Service. The type of each service is defined by the ServiceType enumeration in the metamodel. Tables 4.6, 4.7, and 4.8 show examples of Service instances that can be used in models, listing the typical communication, interaction awareness and message transmission services respectively.

Table 4.6. Communication services (requirements) to support interactions between roles.

Communication Service	Meaning	References
Start audioconference	The actor playing the role involved in the interaction can start an audio conference to interact with the counterpart. Any role can join an audioconference when invited.	(Miranda et al., 2007)
Start videoconference	The actor can start a videoconference with the counterpart.	(Miranda et al., 2007; Oliver et al., 2006)
Open whiteboard	The actor can open a shared whiteboard to interact with others.	(Miranda et al., 2007; Spitzberg, 2006)
Send text message	The actor can send a text message to others.	(Herskovic et al., 2019;

		Miranda et al., 2007)
Send audio	The actor can send audio messages to others.	(Ochoa et al., 2009)
Send video	The actor can send video messages to others.	(Ochoa et al., 2009)
Send image	The actor can send images to others.	(Ochoa et al., 2009)
Send structured data	The actor can send structured data.	(Wu et al., 2012)
Send file	The actor can send files with a predefined (or known) format.	(Bubaš, 2001; Herskovic et al., 2019; Ochoa et al., 2009)

Table 4.7. Interaction awareness services.

Awareness Service	Meaning	References
Incoming messages	It provides awareness information about incoming messages for a user (i.e., messages sent by other users participating in interactions with this user).	(Collazos et al., 2019; Herskovic et al., 2013)
Outgoing messages status	It provides awareness information about the status of outgoing messages.	(Collazos et al., 2019; Herskovic et al., 2013)
Other's presence	It provides awareness information about other users' presence. This is information from a real or potential counterpart in an interaction.	(Herskovic et al., 2013; Pinelle, 2004)
Other's availability	It provides awareness information about other users' availability. Similar to the previous service, this is information from a real or potential counterpart.	(Collazos et al., 2019; Herskovic et al., 2013; Miranda et al., 2007)
Other's location	This is information about other users' location. These users can be local or remote, but they have to participate in the interaction scenario that is modeled.	(Herskovic et al., 2019; Ochoa et al., 2009)
Communication log	It provides awareness information about the communication log (history of exchanges) between the current user and his/her counterpart.	(Dirix et al., 2014; Herskovic et al., 2013;

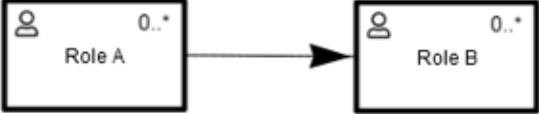

		Miranda et al., 2007)
--	--	-----------------------

Table 4.8. Transmission services to support interactions between roles.

Transmission Service	Meaning	References
Send to one	The role acting as sender can send messages to only one receiver during the interactions with it.	(Miranda et al., 2007; Ochoa et al., 2009)
Send to many	The role acting as sender can send messages to several receivers at the same time during an interaction (it includes the <i>send to one</i> service).	(Miranda et al., 2007; Ochoa et al., 2009)
Send to all	The role acting as sender can send messages to several receivers at the same time in the interaction (it includes both the <i>send to one</i> and <i>send to many</i> services).	(Miranda et al., 2007; Ochoa et al., 2009)
Recipients can reply to sender	Roles acting as message recipients can answer to the sender/transmitter.	(Li et al., 2021)
Recipients can reply to all	Roles acting as message recipients can answer, through a single message, to all actors participating in an interaction (i.e., the message sender and the other receivers).	(Li et al., 2021)
Message priority	The messages include a priority that can be used to determine the order in the delivery process.	(Li et al., 2021)
Message encryption	The messages exchanged during an interaction must be encrypted and decrypted.	(Miranda et al., 2007; Ochoa et al., 2009)

The service directionality can be represented in the language notation. For those interactions in which both roles have at least one service available, the interaction is denoted by a line connecting the roles (as shown in Table 4.9). However, for those interactions in which only one of the roles has at least one service, and the other has none, the interaction is adorned with a black-filled arrow pointing to the receiver role of the service. Table 4.9 shows the notation for service directionality.

Table 4.9. Services directionality.

Directionality	Meaning
	<p>Role A has one or more communication services defined to interact with Role B, whereas Role B has not defined communication services to start an interaction with Role A. Role B can answer the interaction requests delivered by nodes playing the Role A.</p>
	<p>This relationship represents that both Role A and Role B have communication services defined to interact with the counterpart.</p>

4.6. Examples of Interaction Graphs

In order to help the reader understand the structure and information provided by the interaction graphs, next we present a couple of simple examples.

- Uber Single Trip Service -

In order to exemplify the use of some basic elements of the proposed modeling language, let us consider a single interaction scenario from a specific PDCP: *uber single trip*. In this process an uber user (the rider) requests transportation by choosing his/her pickup and drop-off destination through the Uber platform (Uber Company, 2022). Then, the platform sends the information provided by the rider to several drivers near the rider.

Once a driver has accepted to perform the ride, the uber platform assists drivers providing them navigation information about the rider's location, as well as the route to get to the final destination. The platform also supports the riders, showing them information about the driver's location or the estimated time to reach the rider's location. The platform also helps drivers optimize matchmaking with riders and it allows them to interact if required. The rider can prepay for the service using a credit or debit card, or using cash.

Figure 4.2 presents the stakeholders and developer views that specify the interaction scenario using CIMoL. That scenario (model) involves three roles: *rider*, *driver*, and *uber platform*. *Uber platform* is an agent, whereas the remaining roles are human actors. Uber platform is a service that assists the rider and the driver when it is required to help them to make decisions about the trip (for instance cancel the ride).

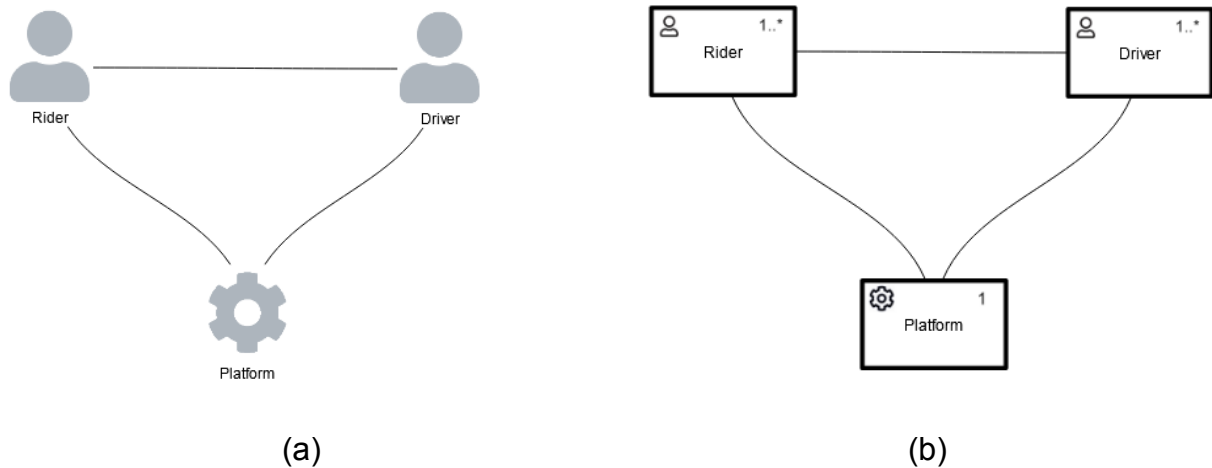


Figure 4.2. Interaction scenario that is part of the *Uber trip* process model: (a) stakeholders view; (b) developers view.

- *Informal Elderly Caregiving Process* -

To exemplify the use of other elements of the proposed modeling language, let us consider a particular PDCP: *Informal Elderly Caregiving*. In this process, families coordinate themselves to take care of their older members. The example is based on the scenarios described in (Gutierrez & Ochoa, 2017), where family members assume implicit roles to fulfill the needs of the older adults. For simplicity, we assume that the mentioned process is constituted by only one phase, and that phase is composed by only a work ambit, which we will name *Elderly Caregiving*.

Figure 4.3 illustrates the stakeholders and the developer views of the interaction scenario using CIMoL. That interaction scenario involves five roles: *emergency service*, *caregiver*, *elderly*, *employee*, and *family caregiver*. *Emergency service* is an agent, whereas the remaining roles are human actors. The emergency service is a healthcare service that the elderly or a caregiver can request by pushing a physical button (like a Panic Alarm Button). Then, a software agent receives the message and derives it to a call center operator who will analyze and manage the request (not modeled in Figure 4.3).

The *employee* is a companion person that assists the elderly during the workday for a salary. *Employees* and *family caregivers* (e.g., children or grandchildren of the elderly) play the role of *caregivers*. Although they have specific roles, both have an *is-a* relationship with *caregiver* (a general abstract role).

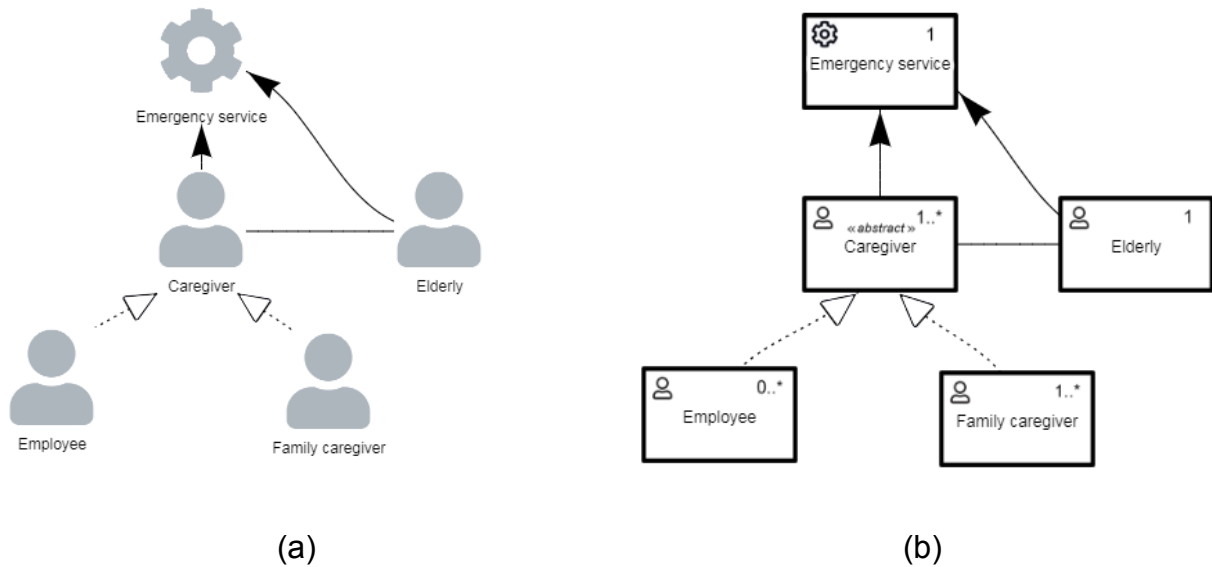


Figure 4.3. Interaction scenario that is part of the *Elderly Caregiving* process: (a) stakeholders view; (b) developers view.

The interaction between *caregivers* and *elderly* is bidirectional, therefore any of them can start an interaction with the counterpart. The interaction graph also shows two unidirectional relationships: *caregiver-emergency service*, and *elderly-emergency service*. This means that both, *caregiver* and *elderly*, can initiate an interaction with the *emergency service*, but not vice versa.

Since *employee* and *family caregiver* are also from the type *caregiver*, both can interact with *elderly* and *emergency service*. Finally, since *caregiver* is an abstract role, it cannot have direct instances, i.e., there are no caregivers by their own, instead, every actor that is a caregiver is either an employee (instance of the role *employee*) or a family caregiver (instance of the role *family caregiver*).

4.7. Summary

In this chapter we have described CIMoL, a role-interaction modeling language designed to specify interactions among actors participating in a PDCP. The language includes two dialects (or views): the *stakeholder view* that is aimed at customers, and the *developer view* that is targeted to software providers (i.e., engineers involved in the development of a collaborative system). In order to illustrate its capabilities, two interaction scenarios corresponding to two PDCP were described.

Chapter 5. Evaluation of the CIMoL Capabilities

This chapter presents an expert opinion study designed to evaluate two aspects of CIMoL. The first aspect is the suitability of the CIMoL concrete syntax; particularly, its potential to be understood by developers and stakeholders. The second aspect is the capability of the language to specify interaction scenarios involved in PDCPs, and derive interaction requirements and awareness services from them, in other words, to determine how well CIMoL addresses the modeling requirements stated in Section 1.3, which were also used to analyze the related works. The evaluation results show where CIMoL contributes to advance the state-of-the-art in the domain of interactions modeling in PDCPs.

This expert-judgment evaluation study took several suggestions from the research conducted by Conboy and Fitzgerald (Conboy & Fitzgerald, 2010), and the evaluation dynamic was based on the Wideband Delphi process (Boehm, 2002). Next, we describe the study settings, the evaluation process of these two aspects, and also a summary that interrelates the study results. The summary analyzes the potential capabilities of interaction graph representations, as instruments to help reach a shared understanding between developers and stakeholders during the requirements engineering process.

5.1. Study Settings

The study involved five stages and a feedback loop (Figure 5.1). In the first stage we defined the instruments used to evaluate: a) the properties of the language's concrete syntax from the perspective of the "physics of the notation" (Moody, 2009), and b) the capabilities of the language's visual notation to represent computer-mediated interaction scenarios.

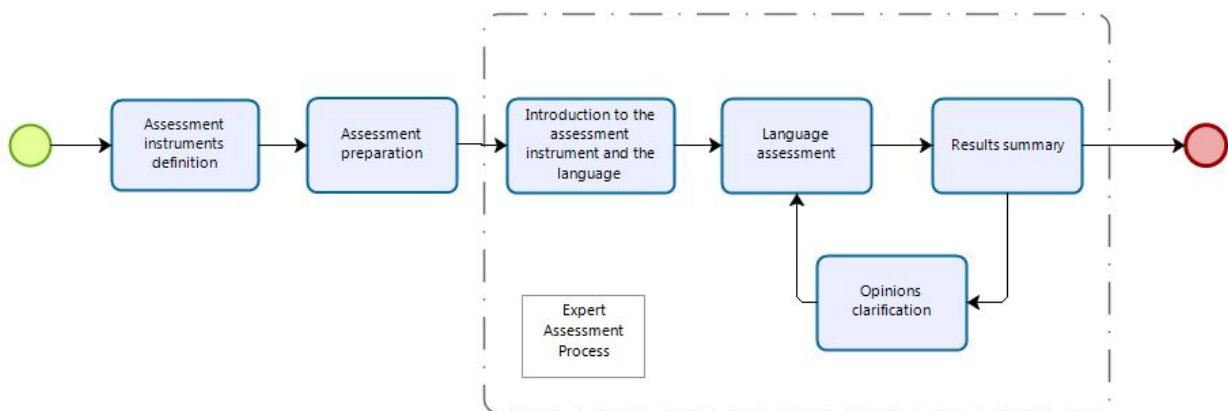


Figure 5.1. Structure of the assessment process.

During the second stage, the documents with the information for the experts were prepared, to help them evaluate the two aforementioned aspects. As part of the expert assessment process, both aspects of CIMoL were then evaluated. Particularly, a first session was held through videoconference, where the experts were explained the purpose of the evaluation process and its dynamics. In that session (third stage of the process), the evaluation instruments were introduced and also the language description. Documents describing the CIMoL's concrete syntax and metamodel were given to the experts. Moreover, a specification of the evaluation process and evaluation instruments were delivered to them.

The experts individually analyzed the information of the language, and they completed the assessment instruments (fourth stage). Their opinions were collected and analyzed to identify discrepancies (fifth stage). In the case of large discrepancies, the consolidated results were delivered to experts, who performed a joint discussion to clarify and eventually change the rates assigned individually (feedback loop). The experts then had to individually complete the instruments again to indicate if they kept or changed their previous opinion (back to the fourth stage).

This feedback cycle is repeated until the distance between the experts' opinions was within acceptable ranges, or the opinions became definitive (no changes on them). This dynamic is recommended by the Wideband Delphi process to align individual assessments among experts (Boehm, 2002). Next, we describe in more detail the activities performed in each stage and their outcome.

5.2. Definition of the Assessment Instruments

We define two instruments in this assessment. The first one is oriented to technically evaluate the physics of the CIMoL visual notation (i.e., its potential understandability for the end users), and the second one is focused on determining the suitability of the language to represent the interaction scenarios that are present in PDCPs (i.e., the CIMoL's expressiveness). Next, we explain each instrument.

5.2.1. Instrument for Assessing the Understandability of the CIMoL Visual Notation

The first step was to review a set of publications reporting the evaluation of visual modeling languages, for instance, the studies in (Weißenberger & Vogel-Heuser, 2012), (Netto et al., 2013), and (Dangarska et al., 2016). Most of them define a questionnaire that is applied after reviewing the concrete syntax of the language, and the items of such an instrument evaluate the physics of the visual notation considering the principles defined by Moody (2009).

Following the same strategy, we defined a set of statements based on the practical use of the design principles for visual notations proposed by Moody (2009). An iterative process was conducted with the support of two researchers, who had extensive experience in design and use of visual modeling languages. The purpose of the exercise was to evaluate and adjust the items of the questionnaire until reaching the final statements. These supporting researchers provided opinions on the usability and relevance of the instrument (through several reviewing rounds), considering their experience and also current literature.

Table 5.1 shows the questionnaire (list of items) defined to evaluate the CIMoL concrete syntax; i.e., its visual notation. Such an instrument deliberately contains some items with a negative and others with a positive connotation. This was done with the purpose of making the evaluators carefully analyze each statement before indicating their level of agreement with the sentence. The answers of the evaluators were expressed in a 5-point Likert scale that considered the following values: 1: *strongly disagree*, 2: *disagree*, 3: *neither agree nor disagree*, 4: *agree*, and 5: *strongly agree*.

Table 5.1. Instrument used to evaluate the CIMoL visual notation.

Item	Description
I1	There is symbol <i>redundancy</i> in the language (there is redundancy when multiple graphic symbols represent the same semantic construction).
I2	The language has <i>no overload</i> of symbols (there is <i>overload</i> when two different constructions are represented by the same graphic symbol).
I3	There is an <i>excess</i> of symbols in the language (there is an excess when there are graphic symbols that do not correspond to some semantic construction).
I4	There is not a <i>deficit</i> of symbols in the language (there exists a deficit when there is a semantic construction that is not represented by some graphic symbol).
I5	Language symbols are clearly <i>distinguishable</i> from each other.
I6	Language symbols use visual representations whose appearance does not suggest their meaning.
I7	The use that the language makes of <i>visual variables</i> (shape, color, size, texture, orientation) is enough to distinguish the elements from each other.
I8	The graphic elements of the language are not <i>enriched with text</i> to aid understanding (i.e., the language does not use text to complement graphics).
I9	The number of graphic symbols of the language is <i>manageable</i> .
I10	The language does not provide mechanisms for <i>modularization</i> (division of a large process into small parts or subprocesses).
I11	The language provides mechanisms for <i>hierarchical structuring</i> (representing processes at different levels of abstraction or detail).
I12	The language has no <i>mechanisms to simplify navigation and transitions</i> between diagrams.

I13	The language has mechanisms to create a <i>complete conceptualization of the system</i> from separate diagrams.
I14	The language does not use <i>different visual dialects</i> for different audiences.

The first nine items (I1 – I9) evaluate the suitability of the set of symbols defined in the concrete syntax for software engineers that have to use it to represent interaction scenarios. The items I10 and I11 assess the capability of the language to manage the complexity of the PDCP specification. The items I12 and I13 evaluate the integration of information from separate diagrams. The last item (I14) assesses the potential capability of the language to be used in practice for different audiences; e.g., developers and stakeholders. A summary of the assessed aspects is shown on Table 5.2.

Table 5.2. Aspects of the language being assessed.

Items	Assessed aspects of the concrete syntax
I1 - I9	Specific features of symbols
I10 - I11	Capability to manage the complexity of the PDCP specification
I12 - I13	Information integration from separate diagrams
I14	Capability to support different audiences

5.2.2. Instrument for Assessing the CIMoL Capability to Represent Computer-mediated Interaction Scenarios

Modeling computer-mediated interaction scenarios during the development of collaborative applications, is an activity that usually requires the joint work between developers and stakeholders. Typically, the former are interested in identifying the participants (user roles) and software services required to support interaction among them (e.g., videoconference or asynchronous messaging), and the latter should provide information about the roles participating in the process, and validate the developers' proposals. Therefore, both must be able to understand and validate the representation of interaction scenarios involved in a PDCP. Such a representation is an agreement between developers and stakeholders about the interaction services that must be included in the collaborative application, considering the participating roles and the interaction needs among them.

The interaction scenarios specified using CIMoL visual notation make the roles of the actors participating in the PDCP explicit, and also the interaction services required by these roles. After a literature review, the author identified seven requirements to address by the interaction model representations (Canché et al., 2022c). These requirements are those explained in Section 1.3, and indicate that these representations should:

1. Specify and characterize roles participating in a PDCP, and also interactions between them.
2. Be understandable for stakeholders.
3. Ease the building of a shared understanding between stakeholders and developers about the scenarios to be supported.
4. Represent the interactions at different abstraction levels.
5. Manage the complexity and size of the interaction scenario specifications.
6. Provide mechanisms for automatic model consistency checking.
7. Derive interaction services from the interaction models in an automatic way.

Using these requirements, a second questionnaire was built to evaluate the suitability of the language to represent the computer-mediated scenarios that are present in PDCPs (Table 5.3).

Table 5.3. Instrument used to evaluate the capability of the CIMoL visual notation to represent computer-mediated interaction scenarios.

Item code	Description
I1	The language allows specifying and characterizing roles participating in a PDCP, and also interactions between them.
I2	The visual models represented using CIMoL are not understandable for stakeholders.
I3	These visual representations allow building of a shared understanding between stakeholders and developers about the scenarios to be supported.
I4	The language does not allow representing the interactions at different abstraction levels.
I5	CIMoL provides mechanisms to deal with the specification of large or complex interaction scenarios.
I6	The language does not provide mechanisms for automatic model consistency checking.
I7	CIMoL allows identifying (or automatically deriving) interaction services from the models of the interaction scenarios.

The first item (I1) determines if the language allows specifying the roles participating in a PDCP, and the interactions among them. The items I2 and I3 evaluate the capability of the language to generate representations understandable to stakeholders. The items I4 and I5 indicate if the language has the capability to divide the product to be specified in parts or views with different abstraction levels. This allows dealing with large and complex interaction scenarios specifications.

Finally, the last two items (I6 and I7) indicate if the models represented using CIMoL allows automatic processing using algorithms; for instance, to validate the models' consistency or derive interaction services from them. The evaluators used this questionnaire and rated each item using a 5-point Likert scale.

5.3. Assessment Preparation

This is the second stage of the assessment process, and it considers two main activities: 1) preparing the documents to be delivered to the evaluators (i.e., the expert researchers), and 2) identifying and inviting the experts to participate in the assessment sessions. Next, we explain these activities.

5.3.1. Documents Preparation

The main goal of this stage was to prepare documents with technical information about CIMoL, to support the experts during the assessment process. These documents were the following:

1. A document with a detailed description of the elements of the modeling language, and a description of the modeling language metamodel.
2. An appendix with visual language design considerations stated by Moody (2009).
3. The process to follow for using CIMoL to model an interaction scenario. The document included an example of an interaction graph that involved roles participating in a PDCP.
4. A document with a description of both the software modeling tool and the prototype generation automatic service, also developed in this thesis work.

5.3.2. Participants in the Assessment

The experts were selected considering their background and expertise in the design, evaluation and use of systems modeling languages. All of them have been working in design and evaluation of visual modeling languages during the last 10 years, and also in development of mobile collaborative systems that support PDCPs. These experts belonged to the following research groups:

- MYDASS Research Group (Modeling and Development of Advanced Software Systems) of the University of Granada, Spain (six researchers).
- IDIS Research Group (Research and Development in Software Engineering) of the University of Cauca, Colombia (two researchers).

The recommended group size for this type of study varies depending on the homogeneity of the sample. Some recommendations indicate 5 to 10 people for a homogeneous population, while 15 to 40 people for a heterogeneous population (Uhl, 1983). This study involved eight experts of a quite homogeneous sample; this is a number within the range of a homogeneous population. The expert-judgment assessment activity offers a good opportunity to use prior knowledge and experience of

the experts to determine the expressiveness and potential usability of a visual language's concrete syntax.

5.4. Introduction to the Assessment Activity

In this activity, that is the third step in the assessment process (Fig. 5.1), we introduce the process, instruments, and supporting information to the participants. The activity involved three meetings, each one with a distinct subset of researchers. The first one involved three experts from the University of Granada, and the second one involved the other three. The third session considered the two researchers from the University of Cauca. The three work sessions were performed during the same week.

In each meeting, the author acted as coordinator and explained the evaluation process, the instruments and the language information to the experts. He also answered questions about the meaning or use of these items. There was also an observer that supervised the process and cared for its soundness.

The meetings were performed through videoconference. After presenting the study context, the evaluation instruments and the language description, all this documentation was sent by email to each expert. The experts had the opportunity to clarify their doubts, both during the meetings and after them.

5.5. Language Assessment Process

This is the fourth stage of the process. In this stage the experts analyzed the material, asked questions, and then evaluated the language using the instruments described in Section 5.2. We next explain the dynamic of this activity and also its main stages.

5.5.1. Dynamic of the Assessment Process

Similar to the dynamic recommended for the Wideband Delphi process (Boehm, 2002), this assessment process also involved three phases, and a feedback loop. In the first phase, or *introductory phase*, the coordinator: 1) informs the participants of the goal and dynamic of the assessment process, 2) explains the technical documents provided to the experts, 3) describes the instruments to be filled with researchers opinions about the language elements and capabilities, 4) answers the experts' questions regarding information about the documents, the instruments and the assessment process.

Then, in an asynchronous phase, 5) the experts explored the concrete syntax analyzing the documents and using the modeling tool to create an interaction model, 6) they had the chance to clarify the doubts before doing the assessment by asking the coordinator, and 7) they filled the instruments and sent them to the coordinator. The experts did not

have a particular deadline to submit their answers, but all of them took between 2 and 4 days to send their first opinion.

Later, 8) the coordinator collected the experts' opinions, condensed and sent them back to the experts. In the case of important discrepancies, 9) the coordinator organized synchronous discussion to clarify the experts' opinions and align positions (when possible).

5.5.2. Exploring the Language

Before evaluating the language, the experts were invited to create an interaction diagram using a visual modeling tool also developed in this thesis work. The goal was to allow them to have a practical experience in the use of the language before assessing it.

Figure 5.2 shows two examples of the modeling tool's user interface. The first corresponds to the "Single Trip" interaction scenario of an Uber process, and the second corresponds to the "Emergency Management" interaction scenario of an Elderly Caregiving process. The visual modeling tool will be introduced in Chapter 6.

The generation of a role interaction graph involved five steps. First of all, the roles (nodes) involved in the collaboration process and their types must be identified. In the second step the designer determines the interactions between nodes (i.e., the relationships). In the third step, the is-a relationships between nodes should be identified, when existing. Fourth, the following features of the model should be specified: optional and mandatory roles, number of participants (per role) required to carry out the collaborative process, user type according to the application use (internal, external, internal/external), and the abstract nodes when required. Finally, the communication, transmission, and interaction awareness services must be specified for each interaction.

Using the modeling tool, the experts created their own interaction scenarios and characterized the roles, interactions, and services according to the guidelines of the modeling process. Then they assessed the language using the instruments described in Section 5.2. Each instrument included two open items; one for reporting strengths and other for weaknesses identified by the reviewers during the assessment process. This allowed participants to transmit their opinions about the CIMoL elements corresponding to specific aspects, as well as their general opinions and suggestions for improving the language.

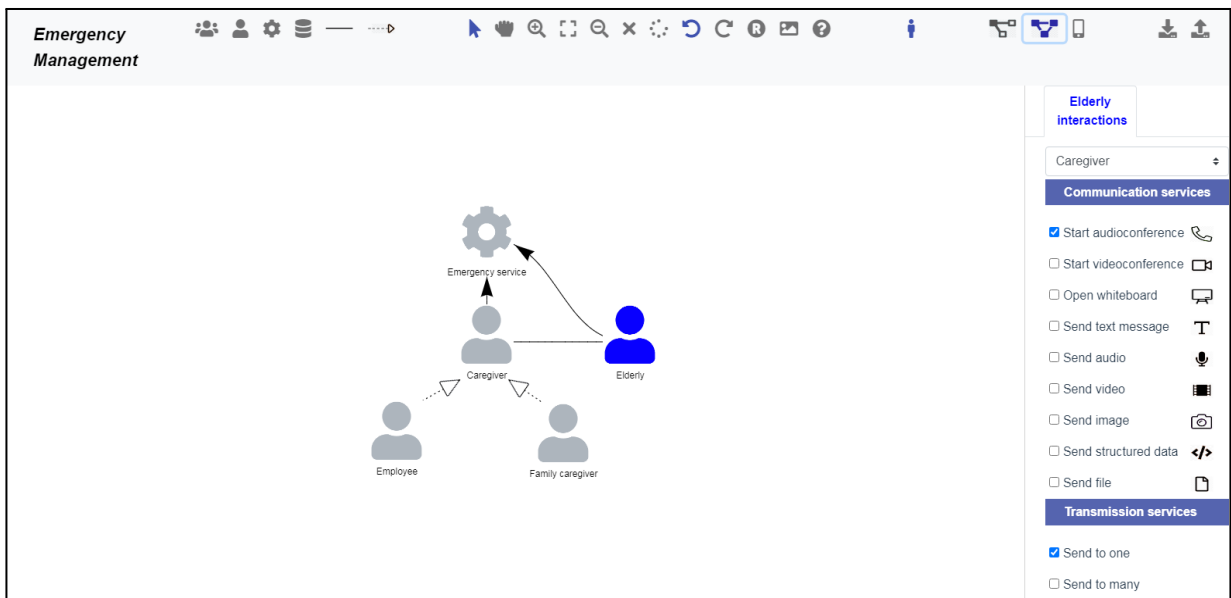
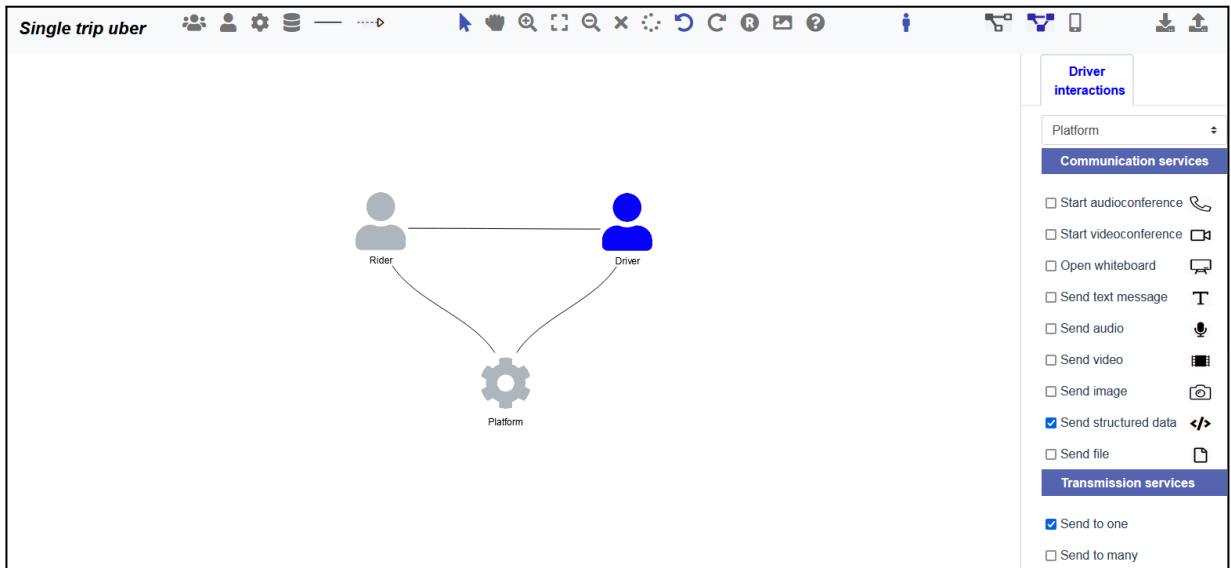


Figure 5.2. User interface of the modeling tool: two examples.

5.5.3. Collecting the Experts' Opinion

The experts sent their evaluations to the coordinator by email. These answers were subsequently reviewed and coded. In case of ambiguity, some clarification was required to the corresponding expert; activity that was performed via videoconference or email no later than one week after receiving the evaluation forms.

For the data analysis, we used recommended procedures for qualitative research, by labeling the concepts and artifacts in such a way as to facilitate the retrieval of information regarding each questionnaire item for each interviewee. An identification code was assigned to the items belonging to the instruments described in Table 5.1 (I1-

I14) and Table 5.3 (I1 - I7), and also to the experts participating in the process (E1-E8), as a way to provide anonymity to the participants.

Subsequently, the data was segmented, since some items to be assessed had a negative meaning, and others had a positive meaning. This was considered to make an adequate interpretation of the answers. Finally, the open comments included in the questionnaires were analyzed separately.

The phase of opinions clarification was not required, because the experts' opinions were clear and their answers were close. The small differences were assumed as normal, as in any expert-based evaluation process.

5.6. Analysis of Results

We now present and discuss the results obtained for each evaluated aspect.

5.6.1. Concerning the Understandability of the Notation

In order to show the results of the understandability assessment in a consistent manner, all items (shown in Table 5.1) were reformulated using a positive sense and keeping the relative values. For instance, if an answer for I2 ("*There is symbol redundancy in the language*") was "*strongly disagree*", this item was changed to "*There is no symbol redundancy in the language*" and the result was represented as "*strongly agree*", keeping thus the relationship between the item and its evaluation result. Specifying the items in that way allows us to get a more comprehensive view of the evaluation results. The following presents the results by evaluation concern.

5.6.1.1. Concerning the Suitability of Symbols set

Table 5.4 shows a summary of the opinions assigned by the experts for the items I1 to I9, which assess the symbols set included in the language, and also the symbols features.

The first three items, which are related to the existence of *redundancy*, *overload*, and *excess* of symbols in the language, obtained mostly positive appreciations (opinions *strongly agreeing* and *agreeing*). There was only a discrepancy in these opinions, since an expert indicated that the notation has some *redundancy* of symbols. Later, this expert clarified his opinion, indicating that in the language description document certain visual elements seem to be synonyms, therefore, he suggested unifying them.

Table 5.4. Score assigned by the experts to items I1-I9 of the first instrument.

Item code	1. Strongly disagree	2. Disagree	3. Neither agree nor disagree	4. Agree	5. Strongly agree
I1	0	1	0	2	5
I2	0	0	0	1	7
I3	0	0	0	1	7
I4	0	4	1	1	2
I5	0	0	0	1	7
I6	0	1	1	2	4
I7	0	0	0	4	4
I8	0	0	0	3	5
I9	0	0	0	1	7

Four experts considered that the language has a *deficit* of symbols (answer “agreed” in item I4). Among the opinions given by them, these are the most representative for the item:

“According to the scope described in the meeting, the symbols are adequate. If you wanted to increase the scope, you would need more symbols” (E2).

“I consider that some services (symbols), such as receive-open a file / audio / image / etc., would be missing, because I would like to receive awareness that something has been received or opened from my interlocutor” (E1).

“In this regard, I am not clear if the services, for example, the message delivery strategy, could also have a graphical representation” (E3).

“I think that some transmission support services could be represented graphically (send-to-one, send-to-many, etc.). If a role interacts with another using a one-to-many multiplicity, it could graphically put a symbol on the interaction line (visual link) indicating if the communication is done only with one or it can be done with several destinations at the same time” (E1).

“I missed symbols that allow modeling the context or communication means, for instance, shared files or whiteboards” (E1).

All these opinions are reasonable, since people expect that the visual modeling language helps them to represent many and diverse things; therefore, it should consider an important number of graphical elements. However, the decision of not including additional symbols in CIMoL was based on experimental evidence collected in the

evaluation of the first two previous notations; i.e., CIMoN v.1 and v.2, also developed as part of this thesis work (Canché et al., 2019).

Moreover, all experts agreed that the *language symbols are distinguishable from each other* (I5). The item I6 refers to the fact that the language symbols use *visual representations whose appearance suggests their meaning* (four researchers strongly agreed and two agreed), however, one expert disagreed indicating:

“It is not clear to me that using different colors (white, black) and an intermediate one (gray) for the user icon, allows someone to easily identify if it is an external or internal user, or both. I would be more inclined to place the user icon outside the role’s box, or on the edge (for the case of external / internal user). Moreover, this user icon could be represented half in white and half in black, instead of using another color” (E6).

For the same item (I6), another expert indicated to be neither in agreement nor in disagreement; particularly he indicated the following:

“I’m not sure that the symbol representing the repository has a clear meaning for stakeholders, and if the stakeholders will be able to use it properly” (E3).

Such statements are understandable, particularly if we consider stakeholders that are watching the symbol for the first time. However, such a limitation can be overcome with an explanation or training session on the symbols' meaning. There were no further comments on the rest of the symbols; therefore, they were assumed as understandable.

Regarding the item I7 (assessing if *the use of visual variables is enough to distinguish the elements from each other*), all the experts had a positive opinion (four agree, and four strongly agree). This was similar to the opinions obtained in item I9 (assessing if *the number of graphic symbols in the language is manageable*), where one expert agrees and the other seven strongly agree with it.

Concerning item I8 (assessing if the *graphic elements of language* are enriched with text to aid understanding, i.e., the language uses text to complement graphics), three experts answered “agree” and the other five “strongly agree” with it.

5.6.1.2. Concerning the capability to manage complexity in the PDCP representations

Table 5.5 shows a summary of the experts' opinions about the items I10 and I11 that assess the capability of the language to manage complexity in the PDCP model representation.

Table 5.5. Score assigned by the experts to the items I10-I11.

Item code	1. Strongly disagree	2. Disagree	3. Neither agree nor disagree	4. Agree	5. Strongly agree
I10	0	1	0	2	5
I11	0	0	1	2	5

As mentioned before, the items I10 and I11 were rated positively for most experts. Particularly, the item I10 refers to the capability of the language to support *modularization in the interaction models* (five experts strongly agree and two agree). This capability comes mainly from the *hierarchical structure* of the models created with CIMoL (item I11). Five experts strongly agree and two agree that the language has such a capability.

Regarding item I10, there was only one discrepancy in these opinions. One expert indicated that the notation lacks modularization mechanisms. This expert stated that:

“I consider the term ‘process’ is quite broad. Therefore, I think the proposed language does not count on modularization mechanisms for entire processes. Perhaps it is worth explaining it in the item description” (E6).

This opinion can be understood since the item I10 states *“the language does not provide mechanisms for modularization (division of a large process into small parts or subprocesses)”*; i.e., it was expressed in a negative way to avoid influencing the answer, and therefore, it probably was misunderstood by the expert.

Concerning the inclusion of an explanation about the meaning of the process as a modeling concept, we considered such an explanation not necessary, because at the beginning of the assessment study, it was stated that CIMoL aims to model roles and their interactions in people-driven processes; i.e., the language does not model events, decisions nor other elements of a process. Therefore, the CIMoL scope is limited to specify the main structure of a process, when it exists, and then to represent the interaction graph for each work ambit and phase of the process. A similar observation about the term “process” was made by the same expert (E6) regarding item I11 (about the hierarchical structure of a process), whose answer was “neither agree nor disagree”.

5.6.1.3. Concerning the capability to integrate information from different diagrams

Table 5.6 shows a summary of the opinions of the experts for the items I12 and I13. These items assess the capability of the language to integrate information from different diagrams.

Table 5.6. Score assigned by the experts to the items I12-I13.

Item code	1. Strongly disagree	2. Disagree	3. Neither agree nor disagree	4. Agree	5. Strongly agree
I12	0	0	0	4	4
I13	0	0	1	3	4

Regarding the item I12 (that assesses if the language has mechanisms to simplify navigation and transitions between diagrams), four experts agreed and four strongly agreed on it. Moreover, there were three experts that initially answered “neither agree nor disagree”. However, after clarifying this aspect with them, they strongly agreed on this item.

Concerning the item I13 (evaluating if the language has mechanisms to create a complete conceptualization of the system from separate diagrams), seven experts agreed or strongly agreed with that statement. There was an expert that indicated “neither agree nor disagree”, but making the following clarification:

"I consider that the language is conceived and focused on the modeling system interactions, and therefore, it has mechanisms to create a complete conceptualization of the system interactions, starting from separate diagrams ...

It could be necessary to integrate another type of diagram that collects the concepts of the problem domain and the relationships between them. For instance, something similar to a conceptual diagram, like the class diagram of UML" (E7).

Such an opinion reflects what really happens with the use of language. Although we thought that the goal of the language was initially clear for the experts, it is understandable that there may be exceptions.

This language does not intend to generate a complete conceptualization of the system, but rather to represent a complete process based on its phases and work ambits that cover all the identified interaction scenarios: i.e., the interactions in the whole PDCP.

5.6.1.4. Concerning the Capability to Support Different Audiences

Table 5.7 shows a summary of the opinions assigned by the experts for the item I14, which assesses the capability of the language to support different audiences.

Regarding this item (I14), which specifically evaluates if *the language uses different visual dialects for different audiences*, most experts rated their opinions positively (two experts answered that they *agreed* and five thought that they *strongly agreed*). This

indicates that the language achieves the capability described. One researcher answered “neither agree nor disagree”, justifying his decision with the following explanation:

“From my perspective, the stakeholders must have previous knowledge to participate. For instance, the logic of the relationships among roles is not general knowledge, but anybody has to understand it to validate a model or help build it” (E4).

According to this expert, the language should not have a dialect targeted to stakeholders, since when carrying out the requirements analysis, they must already have a prior knowledge of the visual nomenclature to be used.

Table 5.7. Score assigned by the experts to the item I14.

Item code	1. Strongly disagree	2. Disagree	3. Neither agree nor disagree	4. Agree	5. Strongly agree
I14	0	0	1	2	5

5.6.1.5. Concerning the Open Comments

Finally, in the items for open comments, interesting suggestions were obtained to consider for future work, for instance:

“It would be interesting to evaluate the suitability of the language to represent large or complex PDCP processes and interaction scenarios, in which many roles with many iterations must be represented in the diagrams. I don't know if the language properly allows software designers to elaborate large or complex diagrams” (E1).

“CIMoL should integrate other types of diagrams as complement; particularly, those that collect the concepts of the problem domain and the relationships between them, something similar to the representations in the UML class diagram” (E7).

“It would be interesting to identify several cultural settings to determine the suitability (understandability) of the language” (E4).

Although these results are still preliminary, they suggest that the CIMoL elements are suitable to perform the representation of interactions between roles in collaborative processes driven by people. Overall, the feedback was positive and constructive, and all experts concluded that the language can be used in practice to model PDCPs and their interaction scenarios. It remains pending the evaluation of its suitability for modeling large and complex PDCPs.

5.6.2. Concerning the Suitability of CIMoL to Specify Computer-Mediated Interaction Scenarios

In order to determine the suitability of CIMoL to represent computer-mediated interaction scenarios we used the instrument reported in (Canché et al., 2022c), which was described in section 5.2.2. Using such an instrument the experts evaluated the capabilities of the language using a 5-point Likert scale. Table 5.8 presents the obtained results.

Table 5.8. Score assigned by the experts to the language in the second assessment.

Item	1. Strongly disagree	2. Disagree	3. Neither agree nor disagree	4. Agree	5. Strongly agree
I1 – Capability of specifying roles and interactions	0	0	0	1	7
I2 – Understandability of models for stakeholders	0	0	1	3	4
I3 – Capability of building a shared understanding	0	0	1	3	4
I4 – Capability of representing interaction at different abstraction levels	0	0	1	4	3
I5 - Mechanisms to deal with large or complex specifications	0	0	1	3	4
I6 - Mechanisms for automatic model consistency checking	0	0	0	3	5
I7 – Mechanisms to identify (or derive) interaction services	0	0	0	2	6

The experts' opinions were quite consistent, and indicated that CIMoL addresses quite properly the seven requirements specified in section 5.2.2. Particularly, the language allows representing roles and interactions among them (I1). The resulting models seem to be understandable for stakeholders (they do for engineers) (I2); therefore, it should be feasible to build a shared understanding between both parts (I3) regarding the interaction services to be provided by the supporting system. However, one expert correctly indicated that they are just envisioning the stakeholders opinion, and therefore, these aspects must be evaluated with stakeholders.

Items I4 and I5 explore the capability of the language to address large or complex models during the product scoping process. Although the experts had a positive feeling about that, they indicated these aspects should be explored using real scenarios with real users.

Item 6 shows how valuable the automatic model checking supported by CIMoL is. Although the results are encouraging, a couple of researchers indicated that this capability is accessible only (or mainly) by a software modeling tool.

Concerning the last aspect, (I7) the capability of identifying (or deriving) interaction services from the models, this aspect is not required like in other similar languages, such as MCM (Herskovic et al., 2019) or CIMoN (Canché et al., 2019). This happens because the engineers and stakeholders have to agree, during the interaction scenarios design time, what interaction services will be considered to support communication between every couple of roles. Therefore, this aspect is addressed, but not through a service derivation mechanism like in other modeling languages.

5.6.3. Comparing CIMoL with the Related Work

This study focused on evaluating the concrete syntax of CIMoL with the judgment of experts in visual modeling languages. The results show a diversity of opinions regarding specific graphic symbols considered in the visual language. However, and based on the opinions and suggestions given by these experts, we can envision that the language positively contributes to reaching a shared understanding between stakeholders and engineers, who participate in the analysis and design of systems supporting PDCPs, which is one of the main goals of this dissertation.

The main limitation found is that there is a potential deficit of symbols in the concrete syntax of the language, since the software services cannot be displayed in the graph, but keeping its understandability for designers and stakeholders. The experts indicated that designers would prefer to have more visual elements in the graph representing such services (i.e., more expressiveness), but not the stakeholders. Therefore, the challenge here is to find a balance between expressiveness and understandability of the language for both parties, i.e., stakeholders and designers.

Considering the empirical evidence reported by Moody (2009), we understand that adding more elements to the language would lead to an overloading of symbols, which jeopardize the understandability of the diagrams. Therefore, we decided to keep the set of symbols and features reported in Chapter 4.

We can also conclude that the language is more expressive than the developed software modeling tool (presented in the next chapter), since the latter includes a set with the current interaction and awareness services being supported by the automatic prototype generator. However, in the future such a tool will add new services that are supported in the language, but not in the current implementation of the modeling tool.

It is also clear that there were some deficiencies in the study, particularly regarding the consideration of synonyms in the terminology. However, these terms were identified and unified to avoid misunderstandings in their future interpretation.

Table 5.9 presents a comparative assessment based on the answers of the researchers about the notations and modeling languages presented in Section 3.2 for the related work, and Section 5.6.2 for CIMoL, considering the design challenges stated in Section 1.3 of this dissertation.

Table 5.9. Modeling aspects covered by the PDCP modeling languages.

Notation \ Aspect	(1) Capability to specify and characterize roles and interactions between them	(2) Capability to be understandable for stakeholders	(3) Capability to achieve a shared understanding of the specifications	(4) Capability to represent the specifications at different abstraction levels	(5) Capability to manage the size and complexity of the specifications	(6) Capability for automatic model consistency checking	(7) Capability for automatic derivation of software requirements
CMMN (OMG, 2016)	Limited	Limited	Limited	Yes	Yes	Limited	Limited
BPMN Plus (Allah Bukhsh et al., 2019)	Yes	Limited	Limited	Yes	No	Limited	Limited
BPMN4SBP (Ben Hassen et al., 2019)	Yes	Limited	Limited	Yes	Yes	Limited	Limited
Little-JIL & hADL (Dorn et al., 2014)	Yes	Limited	Limited	Yes	Limited	Limited	Limited
MCM (Herskovic et al., 2019)	Yes	Limited	Limited	No	No	Yes	Yes
IoT Modeling (Monares et al., 2014)	Yes	Limited	Limited	No	No	Limited	Limited
Collaboration Graphs (Hawryszkiewicz, 2005, 2009)	Yes	Limited	Limited	Yes	Yes	No	No
CIMoN (Canché et al., 2019; Canché & Ochoa, 2018)	Yes	Yes	Limited	No	No	Yes	Yes
CIMoL	Yes	Yes	Yes	Yes	Yes	Yes	Yes

The labels in the table mean: ‘Yes’ - the notation addresses the challenge; ‘No’ - there is no evidence that the notation addresses the challenge; and ‘Limited’ - the challenge is partially addressed. The first five challenges should be dealt with the modeling

language, and the last two with the software tools that support the modeling. As such a table illustrates, no language fulfills all the aspects, except CIMoL, considering the previously mentioned concerns.

As mentioned before, CIMoL addresses the last two aspects, *capability to perform automatic consistency model checking*, and *capability to conduct automatic derivation of interaction services*, through the use of CIMoL modeling tool that is explained in the next chapter.

5.6.4. Threats to Validity

Internal validity: Some factors may influence this study. Among them we can mention the following. First, participants' selection, given that such a process was performed with an invitation to group leaders, some experts there may be unwilling (or less motivated) to participate. Second, dialog capacity and experts' feelings, because virtual participation might have negatively influenced the study. Third, there may have existed some differences between the knowledge level of the experts about visual modeling languages and notations. However, the diversity of perceptions and suggestions helped to enrich the assessment and decrease a bias differently to count on few opinions. Finally, a couple of items in the evaluation of the suitability of CIMoL to represent computer-mediated interaction scenarios (Table 5.8) refer to the stakeholders' opinion. Therefore, the suitability envisioned by the experts to help reach a shared understanding between developers and stakeholders need to be verified empirically with stakeholders.

External validity. We are aware that the participation of the author of this dissertation as guide of the study may have biased the obtained results. However, the participation of such a person comes down to explaining the components of the language and clarifying its meaning if required. We are also aware that the number of participants poses a threat to the generalization of the results, however, we consider that the participation of two expert groups increases the confidence level regarding this aspect.

Reliability. To decrease this threat to validity, a protocol was designed by the thesis author and reviewed by two additional researchers. Also, the collected data was analyzed and discussed by three researchers, spending enough time with each suggestion or comment to unify judgments in the interpretation of each one.

5.7. Summary

In this chapter, an expert opinion study was described, which was designed to evaluate two aspects of CIMoL: (1) the suitability of its concrete syntax and (2) the capability of the language to specify interaction scenarios involved in PDCPs, and derive interaction requirements and awareness services from them. Two instruments were designed to

carry out the assessment of these complementary aspects of the language. Afterwards, such instruments were filled by the experts to collect their answers and analyze them.

Regarding the suitability of the concrete syntax of CIMoL, the results show mostly positive appreciations. Indeed some experts indicated that the notation has a lack of symbols, however it seems reasonable because of the varied elements of the domains' diversity in the applications' development. Concerning the capability of the language to specify interaction scenarios involved in PDCPs the results indicated that CIMoL addresses quite properly the seven requirements specified in Section 5.2.2 (i.e., the language allows representing roles and interactions among them, the resulting models seem to be understandable for stakeholders, etc.).

Based on this assessment study given by the experts, we can foresee that the language positively contributes to achieve a shared understanding about the interaction and awareness services to be embedded into a system supporting a PDCP being developed, which is one of the main goals of this dissertation.

Chapter 6. CIMoL Modeling Tool

Counting on a modeling language is not enough to allow developers and stakeholders to create a shared understanding about the interaction and awareness capabilities of the collaborative application to be developed. In fact, and due to usability reasons, these languages are frequently used through a modeling tool that eases the modeling process and provides persistence to the models.

Therefore, a modeling tool was developed to assist engineers to model and validate the interaction scenarios, considering the concepts specified in the previous chapter. This tool, named CIMoL Modeler, includes an automatic prototype generator that helps stakeholders and developers validate the interaction and awareness requirements. The following sections present the requirements and the main services and user interface of this modeling tool. We then indicate *where*, *who*, and *how* to use CIMoL Modeler to support the stated process.

6.1. Requirements for the Modeling Tool

Based on the study of the role-interaction modeling proposals presented in Chapter 3, the challenges described in Section 1.3, and also the structure of the CIMoL language, we defined a set of core capabilities that should be addressed by the visual modeling tool. These capabilities are the following:

- To allow building role-interaction models considering the structure and concrete syntax of CIMoL. This includes creating and maintaining several models and views: the PDCP structure, the interaction graphs and the system prototypes.
- To allow managing phases and work ambits of a PDCP to reduce its modeling complexity. It considers functional requirements like:
 - a. To create, copy, modify, and remove processes.
 - b. To create, copy, modify, and remove phases and work ambits.
- To support the specification of the diverse actors participating in the PDCP.
- To allow the characterization of each actor type (role), indicating:
 - a. The role name.
 - b. Its mandatoriness, i.e., if the role is optional and mandatory.
 - c. Number of instances required for each mandatory role.
 - d. The role type; i.e., if the actor(s) for each role is internal, external, or both regarding the use of the application to be developed.
- To allow defining the interactions among the roles participating in the process.
- To allow defining an “is-a” relationship among roles, to specialize role types when required.
- To allow specifying the typical interaction and awareness services required between each pair of roles.

- To consider the design principles from the “physics of notations” theory (Moody, 2009), as a way to facilitate the building of shared understandings among developers and stakeholders.
- To perform the automatic validation of the models, i.e., to check the consistency of models.
- Counting on typical capabilities a visual modeling tool; for instance:
 - Copy elements
 - Paste elements
 - Delete elements
 - Undo last action performed
 - Redo last action performed
 - Organize the elements from a model
- To generate prototypes for each human role of the model, to facilitate the validation of the interaction and awareness requirements by developers and stakeholders.

6.2. Structure of the CIMoL Modeler

As mentioned, CIMoL Modeler is a visual and collaborative tool that uses the CIMoL language (Canché et al., 2021), and allows designers to manage four views of an interaction model (see Figure 6.1): 1) phases and ambits (process structure), 2) roles and interactions, 3) interaction and awareness services, and 4) prototype of the mobile system.

The *‘phases and ambits’ (process structure) view* is the higher level of the process representation. It identifies and relates the phases and ambits involved in the interaction process to be supported by the mobile collaborative application. A phase corresponds to a stage (or macro-activity) of the process being supported, and an ambit corresponds to a concern of such a macro-activity. For instance, in hospital work a phase could correspond to *treatments* (a macro-activity), whereas an ambit could correspond to *therapy*, which involves interactions among several actors playing specific roles in that ambit.

The *‘roles and interactions’ view* corresponds to an interaction graph (or interaction model), where the nodes are the roles played by the participants in the process, and the edges represent the interaction capabilities among them. In this view the modelers (i.e., developers and stakeholders) characterize the roles and interactions indicating, for instance, the minimal cardinality of roles to perform an interaction and the generalization relationship between roles.

The *‘interaction and awareness services’ view* provides a more detailed view of the interaction graph, where it is possible to specify the particular interaction and awareness services provided to each role. The tool shows a comprehensive list of these services

(defined in (Canché et al., 2022a)), therefore, the developers and stakeholders should only select the services they want to include in the mobile application, and the tool will use such information to generate the system prototype.

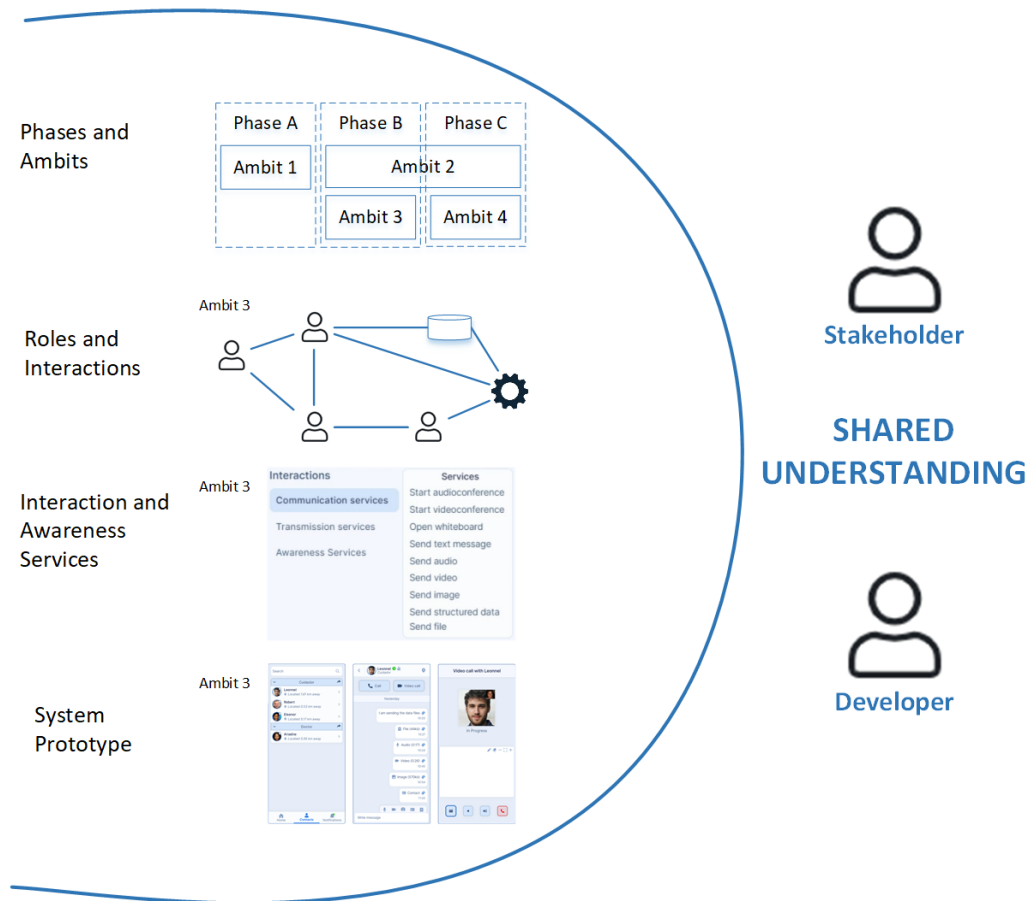


Figure 6.1. Views supported by CIMoL Modeler.

The ‘*system prototype*’ view presents a set of mockups of the mobile system, automatically generated using templates and the information from the process and interaction models. The mockups are clickable, providing thus a quite representative perspective of the product from the end-user perspective. We next describe these views in more detail.

6.3. Modeling the Process Structure

Figure 6.2 presents the main user interface of the processes management service. This interface shows the existing collaborative processes, and allows designers to manage them. The provided functionality includes creating new processes, opening a selected process, as well as renaming, copying, and deleting existing processes.

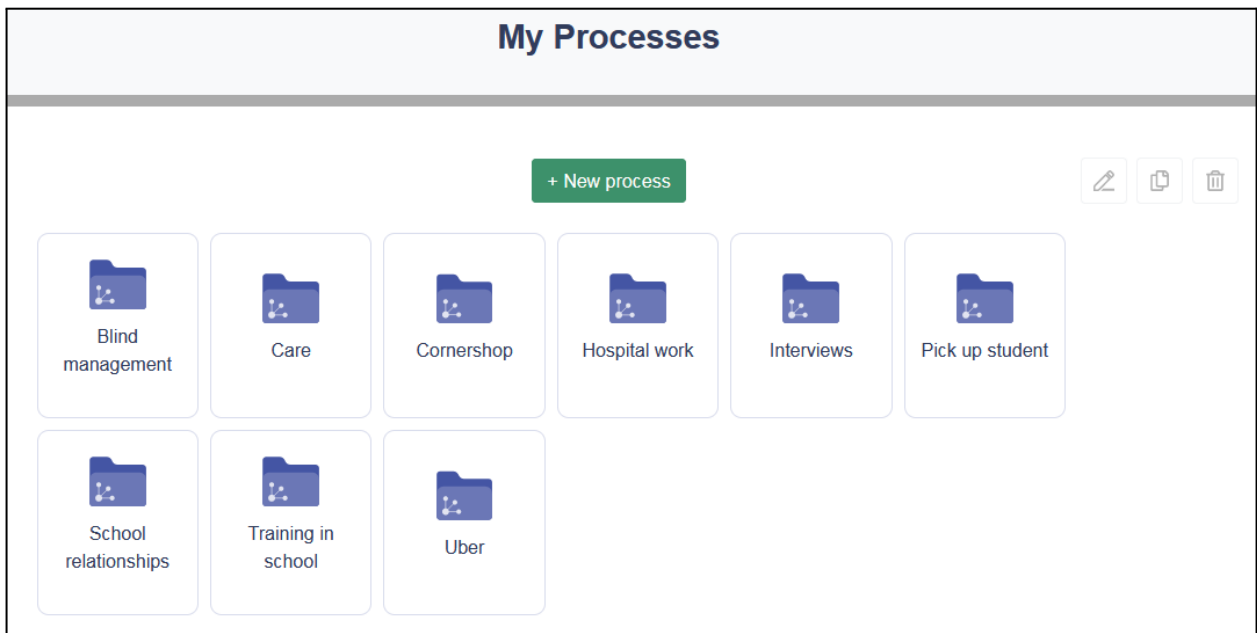


Figure 6.2. Main user interface of the processes management service.

When a process is edited, CIMoL Modeler presents the interface to the users allowing them to manage phases and work ambits. For instance, Figure 6.3 shows the structure of a process named “*Hospital work*”, which involves four phases (columns), and each one has one or more work ambits (rows).

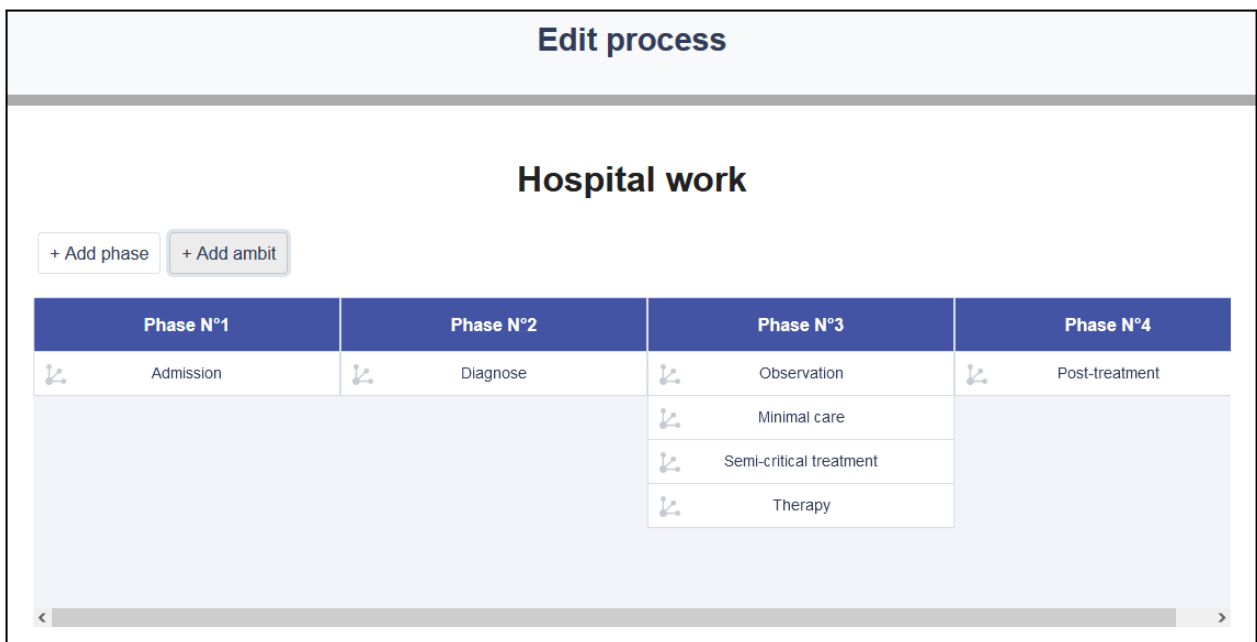


Figure 6.3. User interface to specify phases and work ambits of the *hospital work* process.

The functionality of this interface allows for the following actions:

- *Add phase*: Using the button “+ Add phase” allows designers to create a new phase (Figure 6.3).
- *Rename and delete phase*: Once selected a phase, these actions (rename and delete) are performed through a contextual menu (Figure 6.4).
- *Add ambit*: This allows designers to assign the new ambit to one or more phases, as shown in Figure 6.5. It means that the ambit can belong to more than one phase.

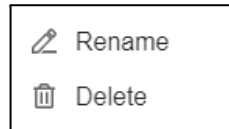


Figure 6.4. Contextual menu for each phase.

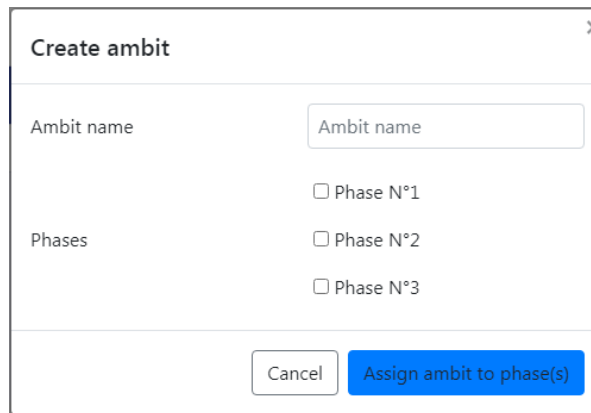


Figure 6.5. Interface for the creation of work ambits.

Each work ambit also counts on the following functionalities, such as shown in the contextual menu of Figure 6.6:

- *View in modeler*: This opens the interaction scenario corresponding to the selected work ambit in CIMoL modeler.
- *Copy ambit*: This functionality allows creating a copy of the selected work ambit.
- *Configure ambit*: This reassigns the selected ambit to one or more phases.
- *Rename ambit*: This allows changing the name of the selected work ambit.
- *Remove from phase*: This removes the selected work ambit from the current phase.

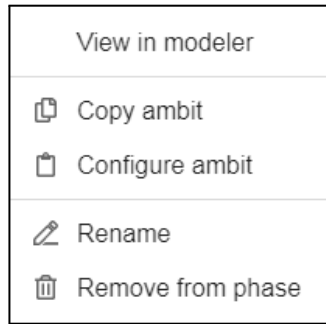


Figure 6.6. Contextual menu for each work ambit.

6.4. Modeling Interaction Scenarios

When the option “*View in modeler*” (Fig. 6.6) is selected, the software tool opens the interaction scenario corresponding to the selected work ambit in the visual modeler interface. An example of this interface is shown in Figure 6.7, which corresponds to a role interaction graph from the work ambit named “*Single trip*” (based on the example from Section 4.6: Uber single trip service).

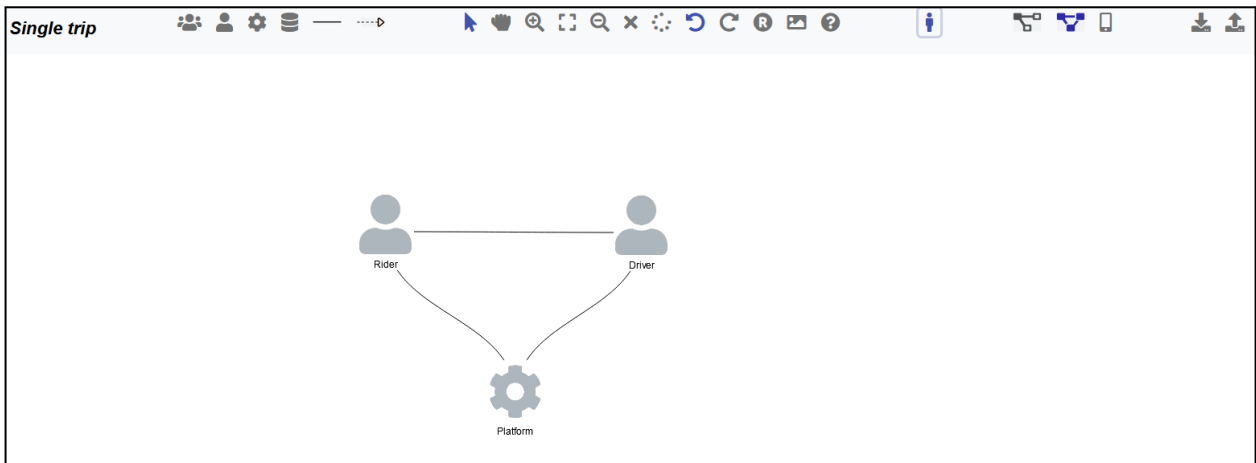
























Figure 6.7. Visual modeler interface (stakeholder view).

To support the model’s management, several options in a toolbar were defined in the visual software tool and introduced in Table 6.1.

Table 6.1. Options in the visual modeler toolbar.

Icon(s)	Function
	Creates a role based on an existing one.
	Creates a human role.

	Creates an agent role.
	Creates a repository role.
	Creates an interaction (link between two roles).
	Creates an "is-a" relationship (relationship between two roles).
	Selects an element.
	Allows panning motion.
	Performs zoom in on the graph.
	Performs zoom fit.
	Performs zoom out.
	Deletes selected elements.
	Performs an auto-arrangement of nodes in the graph.
	Performs an undo of the last action.
	Performs a redo of the last action.
	Makes a report of the interaction services included in the app.
	Downloads an image of the graph.
	Provides support to the user (help).
	Change to stakeholder/developer mode.

	Show interactions with a selected node.
	Show full model view.
	Change to mockup interface.
	Download JSON model (the model is saved in a JSON file).
	Import model (the model is open from a JSON file).

6.4.1. Modeling Roles and Interactions

When a node is selected in this visual model interface, using the *stakeholder view*, the interaction services (i.e., the communication, transmission, and awareness services) can be specified for each interaction from the node. For instance, when the node named “driver” is selected, we can define the services for such a role as shown in Figure 6.8. For simplicity, in this figure only some interaction services are shown.

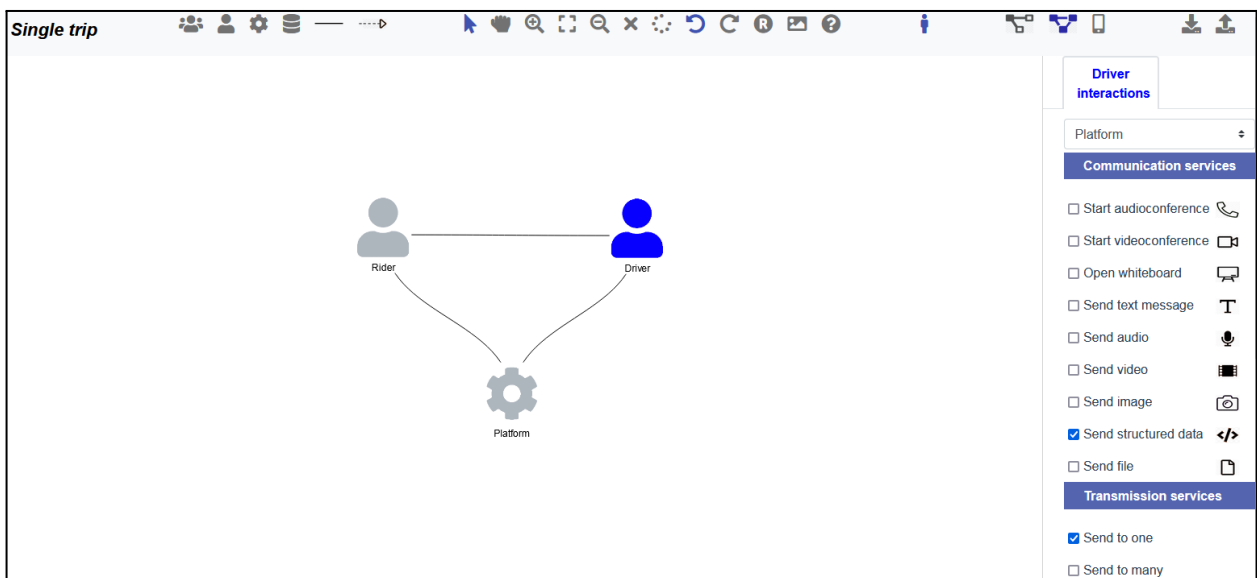


Figure 6.8. Node selection in *stakeholder view*: definition of services for node interactions.

On the other hand, when a node is selected using the *developer view* we can define the node properties, in addition to the services of the node interactions as illustrated in Figure 6.9, which shows the properties for the “driver” role. Concerning the abstract

roles, the properties are slightly different, as shown in Figure 6.10. All the properties for each node type were shown in the description of the language (Chapter 5).

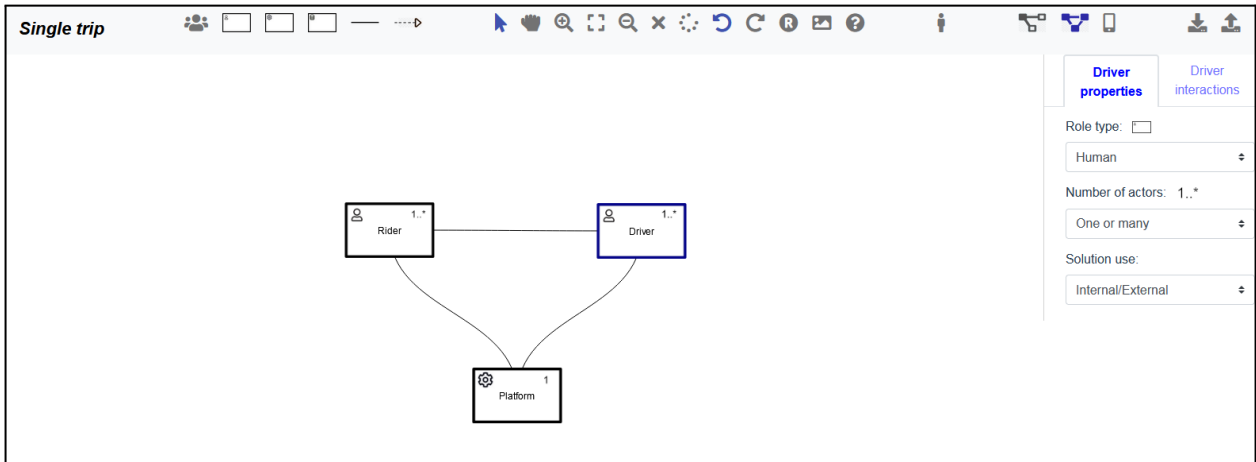


Figure 6.9. Node selection in developer view: definition of properties and services for the selected node.

Role type:

Human

Number of actors: 1..*

One or many

Solution use:

Internal

Abstract role:

No

Yes

Group members:

- Employee
- Family caregiver

Figure 6.10. Configuration of an abstract role.

Regarding the interactions between each pair of nodes, when one of them is selected, the services for each direction can be defined (i.e., the communication, transmission, and awareness services). For instance, for the Driver-Platform interaction, the services for Driver (*Driver* → *Platform*), as well as the services for Platform (*Platform* → *Driver*), can be defined as illustrated in Figure 6.11.

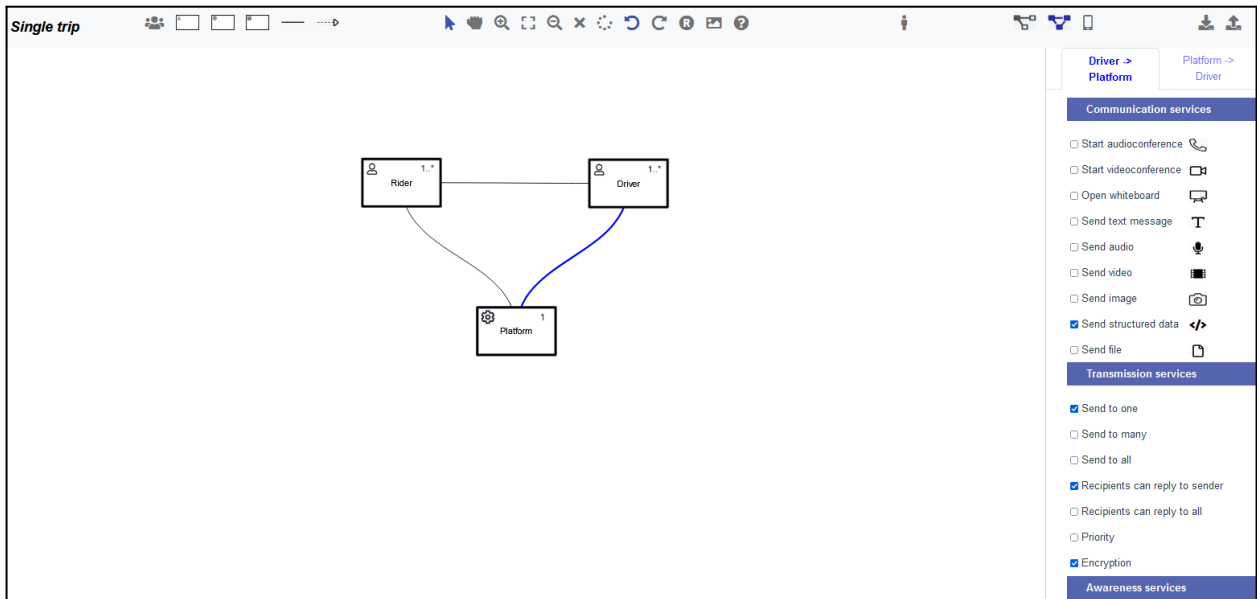


Figure 6.11. Characterization of the selected interaction in the developer view: definition of services for the involved nodes.

6.5. Modeling Prototypes

The software tool includes a module to automatically generate prototypes of the user interfaces to be used by human roles. This helps participants in the definition process to validate the interaction requirements and awareness services to be considered in the development of the mobile application. In such a definition process, software services specified in the interactions for each role are visualized in the form of visual elements. Figure 6.12 shows the main user interfaces of the mockups for the role “*driver*”.

This module contains a role selector in its right side. By selecting a particular role we can visualize its interactions information, as shown in Figure 6.13. Such information can be modified by changing the cells’ value on the table. Any change is immediately reflected in the prototype.

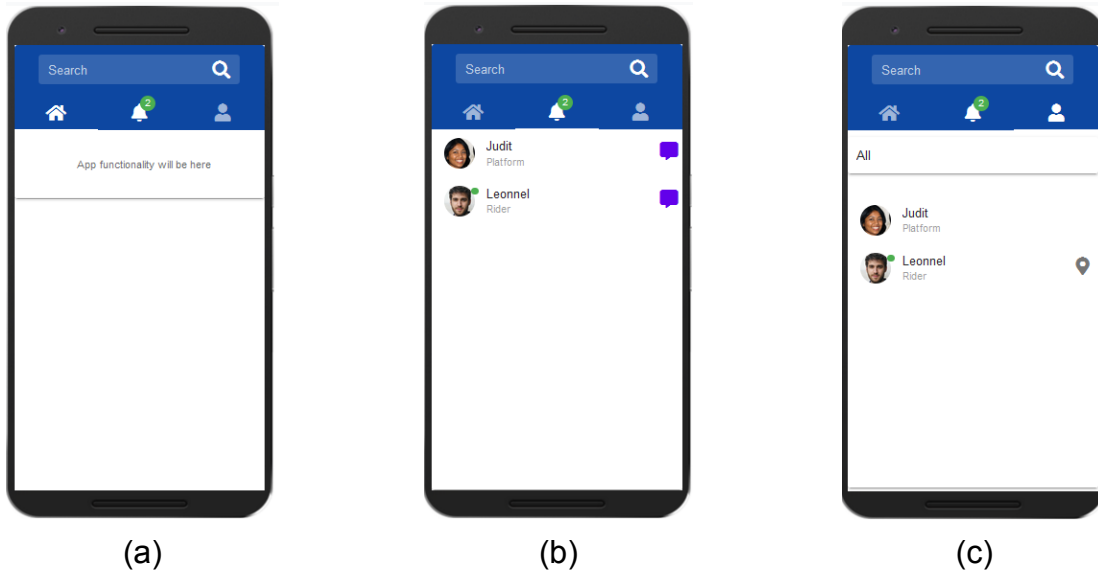


Figure 6.12. Main user interfaces of the mockups for the role *Driver*. (a) Home; (b) Notifications; and (c) Contacts.

Role:

Driver

Communication and awareness services:

Service name	Platform	Rider
Communication services		
Start audioconference		✓
Start videoconference		
Open whiteboard		
Send text message		✓
Send audio		
Send video		
Send image		
Send structured data	✓	
Send file		
Transmission services		
Send to one	✓	✓
Send to many		
Send to all		
Recipients can reply to sender	✓	✓
Recipients can reply to all		
Priority		

Figure 6.13. Interaction information of a selected role.

Next, we briefly describe each section of the mockups.

6.5.1. Home Mockup

The main insight for this mockup is to consider the business functionality and explain to stakeholders why this space exists in the user interface. Since such a functionality is not the aim of this work, we only represent it with the label “*App functionality will be here*” (Figure 6.12.a). For example, if the collaborative application is targeted to *Emergency Management* issues, the functionality will be related to that.

6.5.2. Notifications Mockup

This mockup shows the visual representation of the awareness service, in this case, “*incoming messages*”, which refers to the notification of messages sent to the selected node. For instance, Figure 6.12.b shows the incoming messages for the role “driver”, which include a message from an actor with the role “rider”. That means, the awareness service named “incoming message” for the role “driver” was selected in its interaction with the role “rider”. Additionally, a message icon is shown on this interface (in the right side of each notification), to help designers and stakeholders understand its meaning.

6.5.3. Contacts Mockups

This mockup shows a list of actors playing the roles that can be contacted by actors of the selected node. For instance, Figure 6.12.c shows all the actors whose roles interact with the role “driver”, specifically with actors playing the roles “platform” and “rider”. Additional awareness information can be displayed in the Contacts mockup. Particularly, if the awareness service named “other’s location” (location of other users) is selected for each role that interacts with the current role, a location icon is shown on the right side of the Contacts mockup as shown in Figure 6.12.c.

Besides, this mockup allows designers and stakeholders to select a specific role to show only actors playing such a role. For instance, one can select the role “rider” (Figure 6.14.a) to visualize all the actors performing that role (Figure 6.14.b).

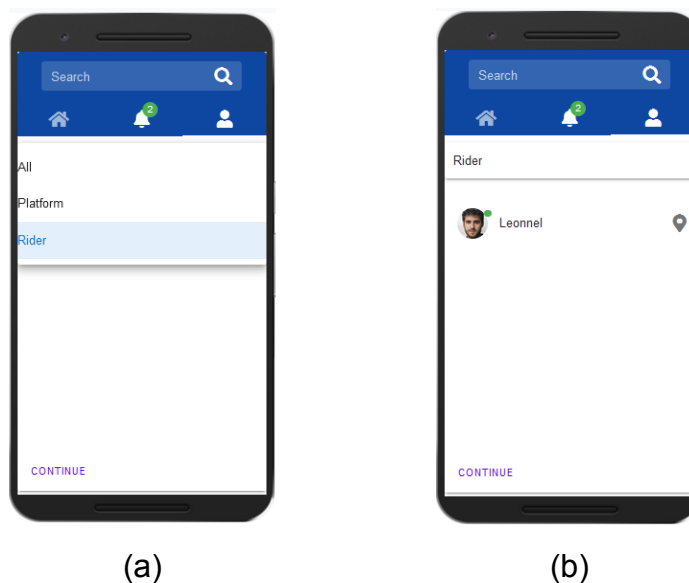


Figure 6.14. Contacts mockup: (a) selecting a specific role; (b) list of actors performing the selected role.

In order to help stakeholders understand how the interaction services can be embedded in an collaborative application, diverse icons were integrated in these mockups, as illustrated in Figure 6.15 (a-b).

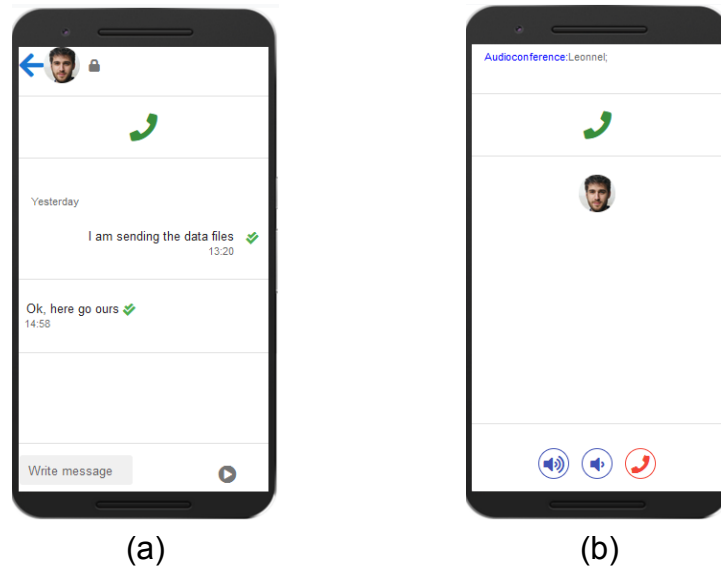
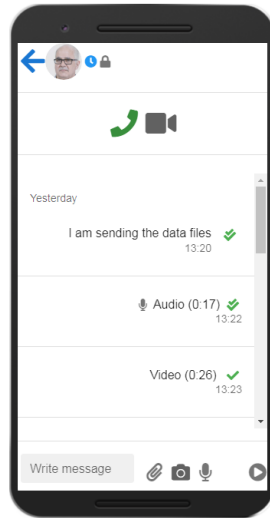


Figure 6.15. Interactions services embedded in the mockup: (a) establishing communication with a specific actor; (b) starting a phone call with the actor

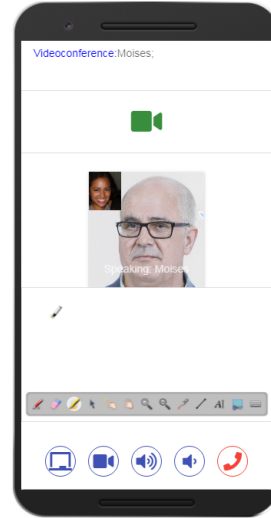
When an actor playing a specific role selects who to interact with, a mockup with the structure of Figure 6.15.a is generated. Here we observe the elements representing the services specified for the interactions in the software modeling tool (for instance *start audioconference*, and *send text message*). Figure 6.15.b shows the particular mockup when a phone call is taken.

Figure 6.16 (a-b) shows other examples of interaction services embedded in a mockup. Specifically Figure 6.16.a illustrates the services *send text message*, and *send audio* among others. Figure 6.16.b shows the result when a videoconference is performed and the whiteboard service is enabled. The meaning of the icons in these mockups is stated in Table 6.2.

Finally, services supporting the transmission (such as, “send to one”, “send to many”, and “send to all”) are represented in the mockups emulating their functionality. For instance, allowing designers to select only one actor if the transmission service named “send to one” is active. For the case of the services “recipients can reply to sender” and “recipients can reply to all”, they are represented if the service “Communication log” is active.









(a)









(b)

Figure 6.16. Visualization of other interaction services embedded in a mockup: (a) establishing communication with a specific actor; (b) starting a videoconference with the actor

Table 6.2. Icons for the interactions services embedded in the mockups of the application.

Icon(s) or Text	Meaning
	Start audioconference
	Start videoconference
	Open whiteboard
I am sending the data files	Send text message
	Send audio message
Video (0:26)	Send video message
	Send image
	Send structured data

	Send file
	Priority
	Encryption
	Incoming messages
	Outgoing messages status
(Circle above the actor image)	Other's presence
(Circle above the actor image)	Other's availability
	Other's location
(Conversation history)	Communication log

6.6. Models Consistency Checking

As described in Chapter 2, *consistency* is a general goal to be obtained while building models, as it contributes to obtain useful and non-contradictory models (Muram et al., 2017). Considering the classification from Spanoudakis & Zisman (2001), CIMoL Modeler uses specialized algorithms to check the model consistency. Such algorithms were designed for analyzing models at design time, while users are specifying the models. The tool helps users create well-formed models, according to the metamodel presented in Section 4.2 and checks the invariants stated for the metamodel. In particular, the tool automatically checks the following rules that help developers minimize errors:

For work ambits:

- Within the process, the name of each work ambit is unique.
- Each work ambit must be in at least one phase of the process.

For nodes (roles played by the participants in the process):

- Within the process, the name of each node must be unique.
- The type of each node must be either: *a human actor, an autonomous agent, or a repository.*
- The number of instances of a node (number of participants playing a role) must be either: *zero or many, zero or one, exactly one, one or many.*

- The solution type (regarding the use of the application to be developed) for each node must be either: *internal*, *external*, or *internal/external*.

For edges representing *interaction between nodes*:

- Each edge connects exactly two nodes.
- Each pair of nodes can have at most one edge connecting them.

For edges representing “*is-a*” *relationships between nodes*:

- Each edge connects exactly two nodes (the nodes must be different).
- Nodes in an “is-a” relationship must have the same type (either human actors, or autonomous agents, or repositories).

6.7. Automatic derivation of interaction services

As described by Turkman & Taweel (2019), providing a systematic derivation of software requirements from the models helps decrease both the derivation effort and human errors, while making this process less dependent on human skills. Furthermore, this automatic derivation of requirements can help both parties to reach an agreement between engineers and stakeholders, and also to define the actual set of requirements to be considered in the system to be developed.

To achieve the third specific goal stated in this thesis, CIMoL Modeler was extended with a software service that uses an interaction model as input, and automatically generates a prototype of the collaborative system to be developed. Since the interaction and awareness requirements were previously specified in the model, they are translated into their visual representations in the prototype.

Such a prototype helps increase the stakeholders' understanding of the software services specified in the interaction models, since there is a correspondence between a *service specified in the interaction model* and a *visual component in the mockups*. We consider that it can also contribute to achieving a shared understanding between software engineers and stakeholders about the scope of the product being developed.

We foresee such a services derivation process is in an advantageous position, compared to those performed by software tools from other languages, such as CIMoN and MCM. When these two languages are used, the requirements derivation process is accomplished by translating the specified model into a potential list of services, which could jeopardize the understanding of the requirements by the stakeholders.

Finally, the use of prototypes eases the identification of adjustments of the software services specified in the model interactions. Any change (adding or removing a service in an interaction) is depicted directly in the prototype, which can help understand the meaning of each interaction service.

6.8. Where the Tool Should Be Used

As any system, the Requirements Engineering (RE) stage of a people-driven collaborative system involves four main activities: *elicitation*, *analysis*, *specification*, and *validation* (Sommerville, 2015) and as known, the main outcome of this stage is the requirements specification of the system to be developed. Using the modeling tool becomes relevant in the activities involved (which are illustrated in Figure 6.17) as follows:

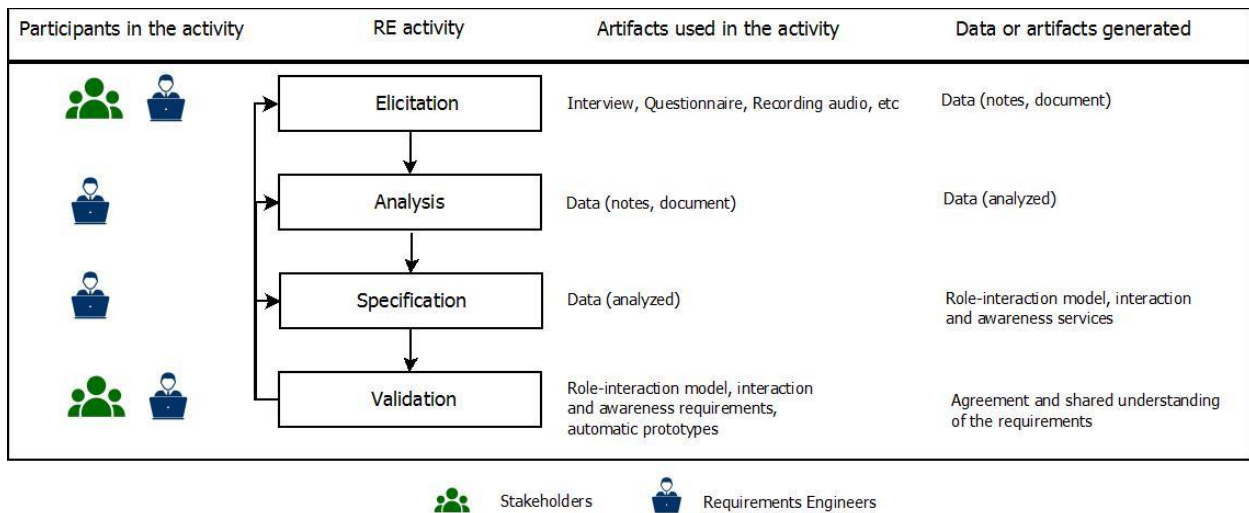


Figure 6.17. Using CIMoL in the requirements engineering stage.

In the *specification* activity, CIMoL Modeler allows designers and stakeholders to generate the role-interaction model and specify the interaction and awareness services. In the *validation* activity, these elements should be also used, together with the prototypes generated by the modeling tool, to create a shared understanding of the interaction and awareness requirements between stakeholders and requirements engineers. These representations can also be used in the *elicitation* and *analysis* activities, as a complement of the artifacts used in them, when the RE stage involves various iterations.

6.9. Who Should Use the Tool and How to Use It

In the development of any system, the activities of the RE stage require the participation of non-technical people (stakeholders) and technical people (typically requirements engineers and developers). Such as shown in Figure 6.17, particular participants are required in the specific activities of RE. For instance, both *elicitation* and *validation* require the two kinds of participants.

As mentioned previously, the only requirements that can be specified at design time by the participants of this stage (in the development of systems supporting people-driven

collaborative processes), are the communication and awareness software requirements (or services). Therefore, it is necessary to identify the interactions of the participants involved in the process, to model them and consequently specify the corresponding requirements for each interaction. In the *specification* activity CIMoL is used to represent these interactions (Figure 6.18).

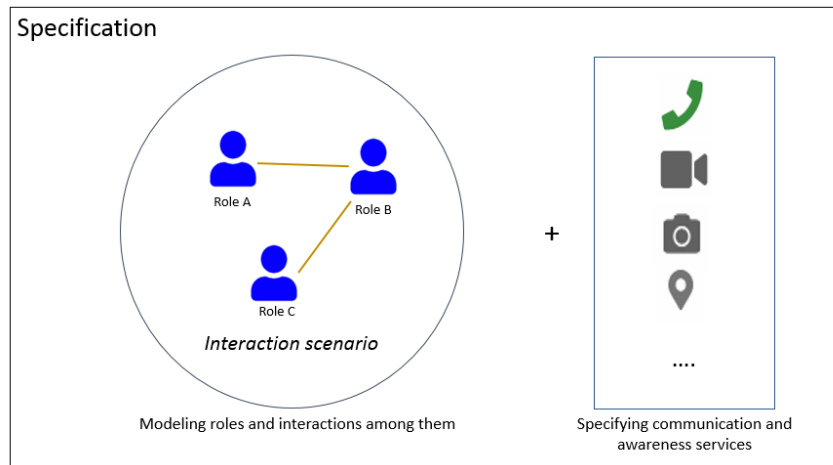


Figure 6.18. Using CIMoL to specify interactions using the modeler.

Although the output generated by the stakeholders and the requirements engineers in the *elicitation* activity can be the interaction scenarios (which include the identification or roles and the characterization of the interactions among them), the requirements engineers should analyze and improve (or adjust) these scenarios adding technical aspects according to the stakeholder needs. Thus, they obtain an accurate set of interaction scenarios at the end of the *specification* activity.

Based on it, a list of potential software requirements (particularly, interaction and awareness services) will be automatically obtained. In the *validation* activity, the last two outputs must be then communicated to stakeholders to get feedback and adjust the services if necessary.

As described in Chapter 2, some artifacts and assessment practices are useful for validating explicit specifications. Artifacts, such as *modeling tools*, and practices such as *prototyping systems*, facilitate the assessment and the achievement of a shared understanding among the participants of the RE stage (Fig. 6.19). Finally, an agreement between stakeholders and requirements engineers is usually achieved.

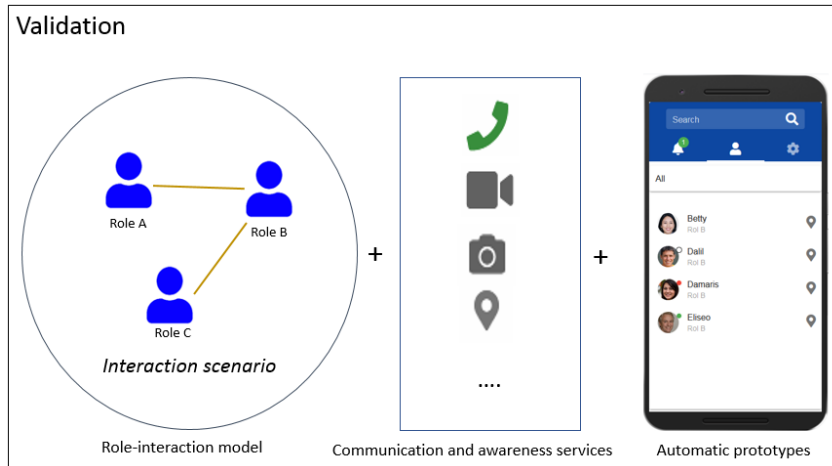


Figure 6.19. The modeler used in the validation activity.

6.10. How to Model an Interaction Graph Using the Modeling Tool

The generation of a role interaction graph involves the steps shown in Table 6.3. First of all, the roles (nodes) involved in the collaboration process and their types must be identified. The second step involves identifying the interactions between nodes (i.e., the relationships). Third, the “is-a” relationships between nodes should be identified (when existing). Fourth, the following characteristics related to the extended notation of CIMoL have to be identified: optional and mandatory roles, number of participants per role required to carry out the collaborative process, user type according to the application use (internal, external, internal/external), and abstract nodes if required. Finally, the communication, transmission, and interaction awareness services must be identified for each interaction.

Table 6.3. Steps to specify a role interaction graph using CIMoL.

Step	Description
1	Identification of roles (nodes) and their types.
2	Identification of interactions (relationships between nodes).
3	Identification of <i>is-a</i> relationships between nodes.
4	Identification of extended-notation characteristics for the nodes: <i>optional</i> and <i>mandatory</i> roles, <i>number of participants per role</i> required to perform the process, <i>user type regarding the application use</i> , and <i>abstract nodes</i> if required.
5	Identification of interaction services (communication, transmission, and interaction awareness services).

6.11. Summary

In this chapter we have described a software modeling tool that assists requirements engineers and stakeholders in specifying and validating interaction scenarios using CIMoL. This modeling tool includes the implementation of two dialects (or views): the stakeholder view that is aimed at customers and non-technical people, and the developer view that is targeted to software engineers involved in the development of a collaborative system.

Also, to ease the validation of the interaction and awareness requirements, an automatic prototypes' generator was included in the tool. These prototypes are generated for roles played by human beings, with the purpose of helping stakeholders and developers to validate and agree the interaction services to be embedded in the application. Finally, to ease the adoption of such a modeling tool, considerations about where, who, and how to use it were presented.

Chapter 7. Case Study: Using CIMoL to Model an Elderly Caregiving Supporting System

To explore the validity of the stated hypotheses, we designed and conducted a case study with the participation of developers and stakeholders. The goal was to define and agree on the scope of a mobile application to support the informal caregiving of older adults; particularly, its interaction and awareness services. Next, we reintroduce the hypotheses to help readers keep them in mind.

In the context of elicitation, specification, analysis and validation of requirements for mobile collaborative applications, this thesis hypothesizes that:

- H1) A domain-specific modeling language (DSML) allows specifying the role-based interaction scenarios between HMUs in people-driven collaborative processes (PDCPs) in a comprehensible way.
- H2) Models specified in the DSML, along with prototypes automatically derived from these models, allow software engineers and stakeholders to reach a shared understanding on the interaction requirements to support PDCPs.

The case study conducted to validate these hypotheses included four replicas; each of them was focused on both:

- Identifying and specifying the interaction and awareness services of a collaborative system that supports the informal elderly caregiving; i.e., a particular PDCP.
- Achieving, together with automatic prototyping, a shared understanding between engineers and stakeholders about these interaction requirements; i.e., the product scope in terms of interaction support.

The evaluation process involved the participation of members of four Mexican families as stakeholders, and eight software engineers as solution providers (i.e., as requirements engineers). The addressed process, i.e., the informal elderly caregiving, is considered a people-driven collaborative process, given the unpredictable nature of the collaboration among participants.

The protocol used in the experiment followed the guidelines reported in (Runeson & Höst, 2008). For our purposes, the unit of analysis was a group formed by engineers and a stakeholder. The latter indicated the older adults' and caregivers' needs, and also expectations concerning the interaction support that should be provided by the mobile application.

As established in the mentioned guidelines, the case and the units of analysis were selected intentionally, in contrast to surveys and experiments, where subjects are sampled from a population to which the results are intended to be generalized.

The following section presents the study design, which includes the research questions, the subjects selection, the data collection procedure, the analysis procedure and validity issues. Section 7.2 describes and discusses the execution of the case study. Section 7.3 reports a discussion of the study results to determine the validity of the stated hypotheses. It also indicates the opportunities and limitations of the proposed language. Section 7.4 shows the threats to validity of this study's results, and Section 7.5 presents a summary of the chapter.

7.1. Case Study Design

In this section we describe the research questions, the selection of cases and subjects, the data collection procedure, and the analysis procedure. We explain each one next.

7.1.1. Research Questions

Considering the stated hypotheses, the main goal of the case study is: *to evaluate the capability of CIMoL to specify the interaction and awareness services to be embedded into a system that supports a PDCP, and also its suitability to help achieve a shared understanding among engineers and stakeholders about these services*. As mentioned, the PDCP used in this case study was informal elderly caregiving. Based on this goal and the work hypotheses, we defined two research questions to answer in the case study. Using CIMoL, through CIMoL Modeler:

- RQ1: Are developers, jointly with stakeholders, able to specify the computer-based interaction and awareness services required by the participants in the informal elderly caregiving process?
- RQ2: Are developers and stakeholders able to achieve a shared understanding on these computer-based interaction and awareness services?

7.1.2. Introduction to the Informal Elderly Caregiving Process

Informal elderly caregiving is a process whose purpose is to provide health and emotional care to an elderly, and assistance with activities of daily living (Plöthner et al., 2019). Typically, these people live at home, whether alone or with family members depending on their physical and mental health condition. This aging paradigm, known as aging in place (National Institutes of Health, 2022), is supported mainly by family members like the spouse/partner, adult-children, and children-in-law (Pinquart & Sörensen, 2011). Sometimes the caregivers network includes friends of the older adult and formal caregivers.

Typically, there is one or more main caregivers (usually the partner and adult-children) that not only assist the elderly, but also keep the track of the process and coordinate the activities of other participants. This process is time-demanding and usually burdens the main caregivers (Gutierrez & Ochoa, 2017), therefore, the participation and coordination of the caregivers' activities play a key role in the cost-effectiveness of such a process. The informal elderly caregiving can be considered a PDCP since, although there are some planned (or routine) activities, the workflow of activities is unpredictable.

7.1.3. Selection of Participants

Eight software engineers participated in the study. All of them had at least five years of experience in analysis and development of software systems. Six of them were located in Santiago, Chile, whereas two of them were located in Yucatan, southern Mexico. Two of them were female and six were male, and the age range was 33 to 38 years old.

Concerning the stakeholders, four regular caregivers of elderly people were involved in the study. All of them were representative of families that collaboratively take care of older adults for at least four years; in some cases, the process was partially supported by external personnel. All of these stakeholders were residents of the Yucatan state, Mexico. All of them were daughters, which is aligned with the information reported in the literature (Barros et al., 2014; Dwyer & Coward, 1991; Gutierrez et al., 2016). The age of the participants was between 35 and 52 years old.

The main criterion used to select the caregivers (stakeholders) was to have recently participated, in an active way, in the caregiving process of an elderly for at least one year. By active way we mean that the person should be in charge of performing activities in favor of the older adult at least three times per week; for instance, cooking, cleaning, providing emotional support, or monitoring the health condition of the elderly. The caregivers should not have mental or sensory impairments, and they should be regular users of smartphones.

In the case of the developers, the selection criteria were mainly two: 1) to have at least three years of experience in analysis and development of software systems, and 2) to have developed at least one collaborative system. After identifying the candidates, they were invited to participate in the study.

To facilitate the subsequent retrieval of the data regarding the participants and ensure their anonymity, an identification code was added to each of them: E1, E2, ..., E8 for engineers, and S1, S2, ..., S4 for stakeholders. Table 7.1 shows the distribution of engineers and stakeholders in each replica of the case study.

Table 7.1. Distribution of participants in each replica.

Participants	Replica
E1, E2, S1	I. Elderly with senile dementia and mobility limitations
E3, E4, S2	II. Elderly with Down's syndrome
E5, E6, S3	III. Elderly with mobility limitations
E7, E8, S4	IV. Elderly with mobility limitations

7.1.4. General Description of the Activity

The developers and the caregivers were individually invited to participate in the conception of a mobile application that supports the informal caregiving of older adults. Developers played the roles of software provider, and the caregivers acted as stakeholders (particularly, users of such an application).

The participants were grouped as shown in Table 7.1. These people interacted through videoconference to specify the interaction requirements of a mobile collaborative application that support the participants in the informal elderly caregiving process. Thus, they tried to create a shared understanding about the interaction and awareness services to be embedded in the application.

Each group performed the requirements elicitation focusing the activity on the identification and validation of interaction and awareness services required by the participants in the caregiving process. Additionally, they participated in an individual semi-structured interview to identify whether developers and stakeholders are able to achieve a shared understanding on these services. All these activities were performed using Google Meet¹.

Two meetings among developers and stakeholders were conducted by replica. In the first meeting, the stakeholders were interviewed by developers to identify the required interaction and awareness services. Such a meeting lasted 24 minutes on average.

In the second meeting, developers explained to stakeholders the interaction model generated with the information gathered in the previous interview. They also showed the mockups of the mobile application that should support the informal elderly caregiving process; which were generated by CIMoL Modeling tool.

The stakeholders were able to validate the interaction scenario that was specified by the developers, and also the services and mockups of the mobile application. This second meeting lasted 17 minutes on average. Figures 7.1 and 7.2 show screen captures of the participants working together in some of these meetings.

¹ <https://meet.google.com/>

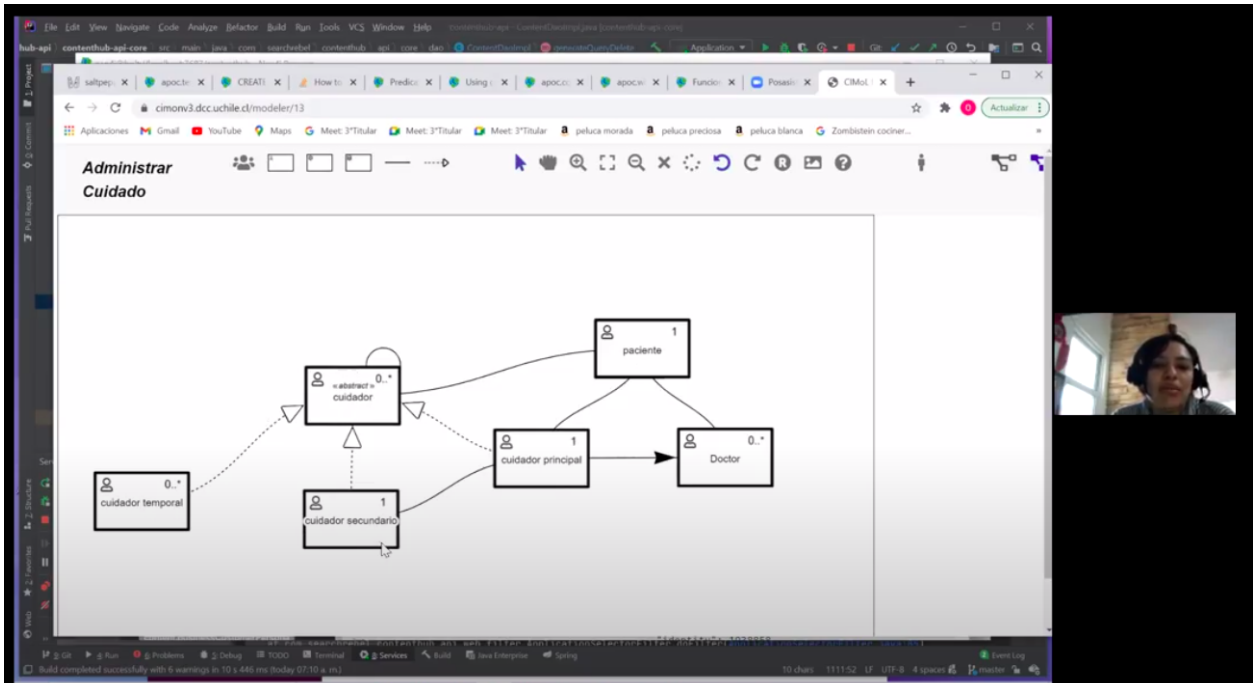


Figure 7.1. Participants defining an interaction scenario in a working session.

Service name	paciente	cuidador
Communication services		
Start audioconference		✓
Start videoconference		✓
Open whiteboard		✓
Send text message		✓
Send audio		✓
Send video		✓
Send image		✓
Send structured data		✓
Send file		✓
Transmission services		

Figure 7.2. Participants validating the interaction and awareness services in a working session.

To carry out the activities described above, the developers had a training session and performed the following activities:

1. Previous to the training session, the developers were informed about the CIMoL language. Particularly, the language description and a guide about how to use it was given to them.
2. At the beginning of the session the CIMoL Modeling tool was presented, including its goals and main components. We also showed examples on how to use it to conduct the requirements elicitation stage.
3. They were then asked to build an interaction model for a specific case, using the CIMoL Modeling tool.
4. Finally, the developers discussed the interaction model and the mockups generated by CIMoL Modeling tool to validate them.

7.1.5. Data Gathering Mechanisms

Several data gathering mechanisms were used in this case study. As suggested in (Runeson & Höst, 2008), it is important to use various instruments to limit the biases eventually introduced by using only one.

Firstly, *observations* were conducted to understand the way in which software engineers perform the requirements elicitation and co-conceive the system and its scope. It allowed us to collect raw data without interfering with the system conception process. The meetings between engineers and stakeholders were recorded in video to then analyze them in detail. An advantage of observations is that they may provide a deep understanding of the phenomenon that is studied (Robinson et al., 2007).

Moreover, *interviews* were used to collect data from the participants once they concluded this requirements engineering process. In these interviews, the dialogue with the subjects were guided by a set of interview questions. In order to ensure confidentiality, engineers and stakeholders were interviewed individually through videoconference. Each interview was customized according to the process carried out by each group of engineers, and thus to collect the suitable data to be analyzed. Moreover, the interviews were also recorded in video with the same purpose.

Finally, the artifacts generated by the participants using CIMoL Modeling tool, were included as complementary sources of information. They were used as supporting elements during the interviews, to determine if a shared understanding between engineers and stakeholders was achieved after these sessions. Additionally, given that the models corresponding to the case study were initially built in Spanish, all of them were translated to English.

7.1.6. Data Analysis Approach

Provided that case study is a flexible research method, and the number of participants is small, a qualitative data analysis approach was used. The main objective of this analysis was to derive preliminary conclusions from the data, keeping a clear chain of evidence that allows us to understand the results (Yin, 2008).

In order to reduce the data processing bias, the analysis was conducted by three researchers to increase the validity of the results. The preliminary results from each individual researcher were merged into a common analysis result.

7.2. Case Study Execution

In this section each replica of the case study is described and discussed. Each of them aims to reconstruct a description of the interactions to be supported by the mobile application to allow family members participate in their particular elderly caregiving process.

During the specification of the interaction model in each replica, the participants were asked to conduct the following activities:

1. Identify the roles participating in the PDCP and their types.
2. Identify the interactions between each pair of roles.
3. Identify the is-a relationships between nodes if required.
4. Identify the optional and mandatory roles, as well as the number of participants per role.
5. Identify the user type regarding the way of using the application (i.e., internal, external, or both types of use), and also identify abstract nodes if required.
6. Identify the interaction services required for each specified interaction.

7.2.1. Replica I: Elderly with Senile Dementia and Mobility Limitations

This family process involved an elderly female with senile dementia and mobility limitations. Senile dementia is a condition characterized by deficits in perceptual, mental, and social functioning, as well as by physical impairments. Patients with this disease have a particular tendency to put themselves at risk in potentially dangerous situations; thus, constant supervision is usually required (Tack et al., 2011).

In addition to the representative stakeholder, who is the youngest daughter and the person responsible for the elder, a sister and a brother who live in the same house contribute directly to the caregiving process. There is also a female employee who

attends three times a week, and is in charge of several activities related to elderly personal hygiene and house cleaning. When necessary, two of the elder's granddaughters assist the family. Finally, a sister who lives in another place sometimes participates in particular caregiving activities. That sister has direct contact with a physician, who usually gives her indications about the manner in which the elder should be treated.

After the requirements elicitation session, the engineers built a role-interaction model which is illustrated in Figure 7.3. This model was used to explain to the stakeholder how the process would be conducted when a system based on roles was deployed. Afterwards, the automatic prototypes generated by the CIMOL Modeler that used this role-interaction model as input, were described to the stakeholder. The prototypes show how the application supports the communication and coordination activities among family members participating in the caregiving process.

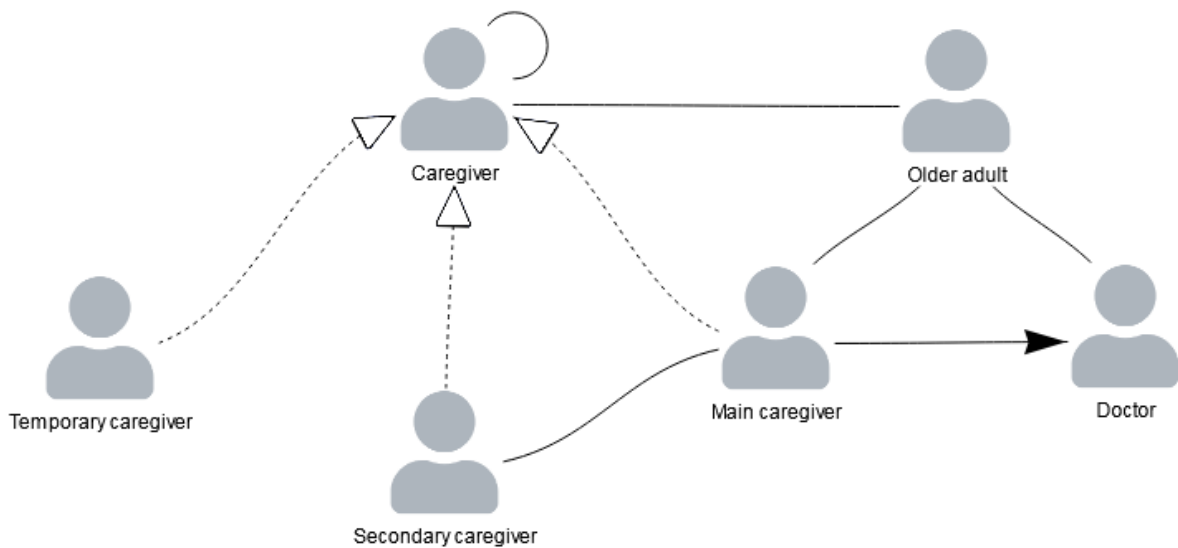


Figure 7.3. Role-interaction model (stakeholder view) specified by the engineers.

Figure 7.4 shows the developer view of the same role-interaction model, built by the engineers, which complements the information described in the stakeholder view (Fig. 7.3).

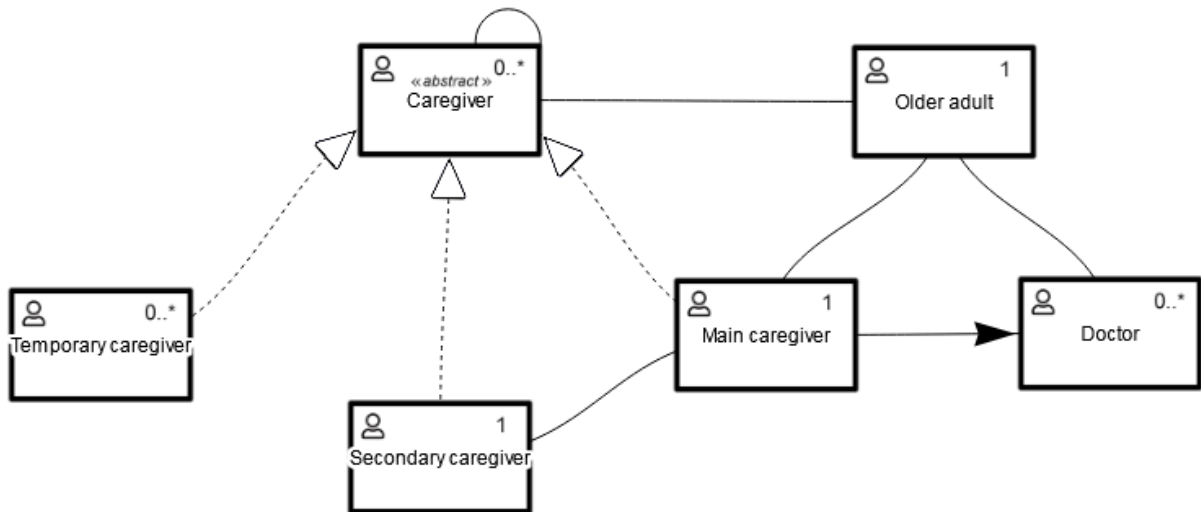


Figure 7.4. Role-interaction model (developer view) specified by engineers.

Additional artifacts are shown in Figures 7.5. The first one indicates the services specified by engineers in the interaction form, corresponding to the interaction from “Secondary caregiver” to “Main caregiver”. The services were established based on the information gathered in the elicitation activity. For setting such an interaction, the user (i.e., the developer or the stakeholder) should click on the line that represents it; after that, the services to support such an interaction can be configured.

<div style="display: flex; justify-content: space-between;"> <div style="border: 1px solid black; padding: 5px;"> <p>Secondary caregiver - > Main caregiver</p> </div> <div style="border: 1px solid black; padding: 5px;"> <p>Main caregiver -> Secondary caregiver</p> </div> </div> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p style="background-color: #4a7ebb; color: white; padding: 2px; text-align: center;">Communication services</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Start audioconference <input checked="" type="checkbox"/> Start videoconference <input checked="" type="checkbox"/> Open whiteboard <input checked="" type="checkbox"/> Send text message <input checked="" type="checkbox"/> Send audio <input checked="" type="checkbox"/> Send video <input checked="" type="checkbox"/> Send image <input type="checkbox"/> Send structured data <input type="checkbox"/> Send file </div>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p style="background-color: #4a7ebb; color: white; padding: 2px; text-align: center;">Transmission services</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Send to one <input type="checkbox"/> Send to many <input type="checkbox"/> Send to all <input checked="" type="checkbox"/> Recipients can reply to sender <input type="checkbox"/> Recipients can reply to all <input type="checkbox"/> Priority <input type="checkbox"/> Encryption </div>	<div style="border: 1px solid black; padding: 5px;"> <p style="background-color: #4a7ebb; color: white; padding: 2px; text-align: center;">Awareness services</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Incoming messages <input checked="" type="checkbox"/> Outgoing messages status <input checked="" type="checkbox"/> Other's presence <input type="checkbox"/> Other's availability <input type="checkbox"/> Other's location <input checked="" type="checkbox"/> Communication log </div>
---	--	--

Figure 7.5. Services to support interactions from “Secondary caregiver” to “Main caregiver”.

Figure 7.6 shows some interfaces of the mobile system prototype, for the role “Secondary caregiver”. The functionality shown in the interfaces allows these users to support communication and coordination with other participants of the process.

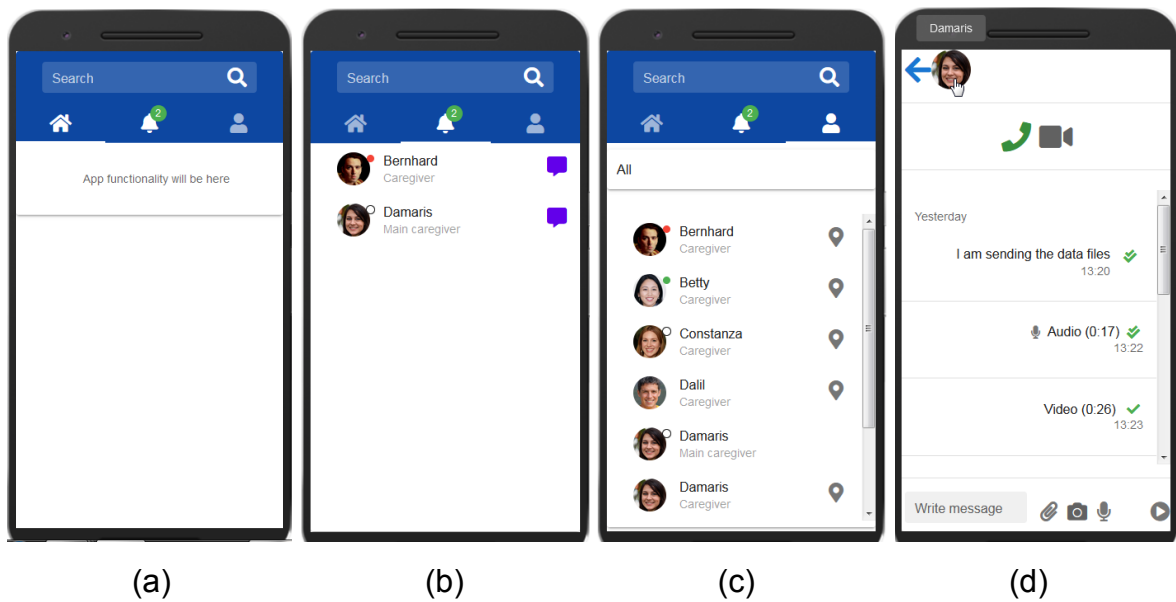


Figure 7.6. Prototype’s interfaces for the role “Secondary caregiver”: (a) home; (b) notifications; (c) contacts; (d) main UI to interact with the “Main caregiver”.

After the explanation of the model, and generating the prototype for the stakeholder, individual semi-structured interviews were carried out to the participants. The instruments to the stakeholders and engineers are detailed in Appendix I and Appendix II respectively.

First, using the corresponding instrument, the stakeholder was asked to identify actors playing the roles identified in the role-interaction model (shown in Fig. 7.3). Moreover, questions related to services required to communicate participants in the process were asked. We then asked if the language allows representing the family needs. Finally, we asked whether the engineers understood the family needs, to identify if both parties were able to reach a shared understanding on the system scope.

All questions included in the instrument (shown in Appendix I) were answered in a correct/positive manner by the stakeholder. Only an issue related to item 2 (do you consider that your sister -the person in touch with the doctor- is represented in some role within the diagram?) was clarified by her:

“Currently, my sister is not in touch with the doctor, but I understand that this diagram represents a future need” (S1).

Such as clarified, she understood that the model proposal is a future work, since the main caregiver has no direct interaction with the doctor. Despite this, she considered that the engineers understood the interaction needs of the informal caregivers.

Regarding the engineers, all instrument questions (shown in Appendix II) were answered in a correct/positive manner by them. For instance, concerning item 9 (do you consider that the prototype helps the stakeholder understand how an application could support participants in this process?) one engineer answered the following:

“Yes, because the prototype is visual, and it contributes to get a better idea in the stakeholder’s mind about how the application will work” (E2).

Regarding the application requirements, they were rated as “very clear” to one engineer, whereas they were only “clear” to the other.

Finally, all the activities established at the beginning of Section 7.2 were addressed by the participants of this replica. However, the engineers made the following mistakes in the modeling process:

- a) Lack of the current interaction between *secondary caregiver* and *doctor*. The engineers justified such a decision with “the future app should be designed to assist the older adult in a direct form, but with this interaction is not possible”.
- b) The interaction between *main caregiving* and *doctor* was added although it is not supported in the real process. In this case, the engineers explained that it represents a future solution.
- c) Possibly, the older adult could not use the software application given its physical condition. Therefore, considering this role could be an error. Further discussion between stakeholders and engineers about it is required.

It is important to remark that the mistakes were presented at the model's version proposed to stakeholders. Provided that the modeling process is iterative, these mistakes were early detected during the first evaluation of the models, therefore the engineers should be able to fix them. The same happens in the replicas reported in the next sections

7.2.2. Replica II: Elderly with Down’s Syndrome

This family caregiving process involved an elderly female with Down’s syndrome. People with this genetic disorder often experience developmental delays, neurological and anatomical linked cognitive impairment, and increased prevalence of medical conditions, such as congenital heart disease and early-onset Alzheimer’s disease (Esbensen, 2010).

Besides the representative stakeholder, who is a daughter and the person responsible for the elder, her husband and children who live in the same house contribute directly in the caregiving process. All caregiving activities were coordinated among them, from feeding to taking the elderly person to the physician (medic).

After the requirements elicitation, the engineers built the role-interaction model shown in Figure 7.7. This model was then used to explain to the stakeholder how the process would be conducted when it is supported by the mobile system. Afterwards, the system prototypes related to this role-interaction model were described to the stakeholder to show how an application would communicate to the family members participating in the caregiving process. The developer view of this model is illustrated in Figure 7.8.

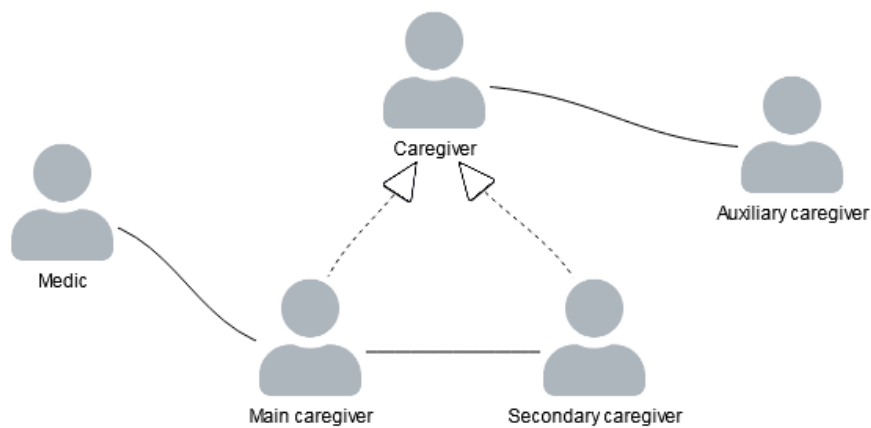


Figure 7.7. Role-interaction model (stakeholder view) specified by the engineers.

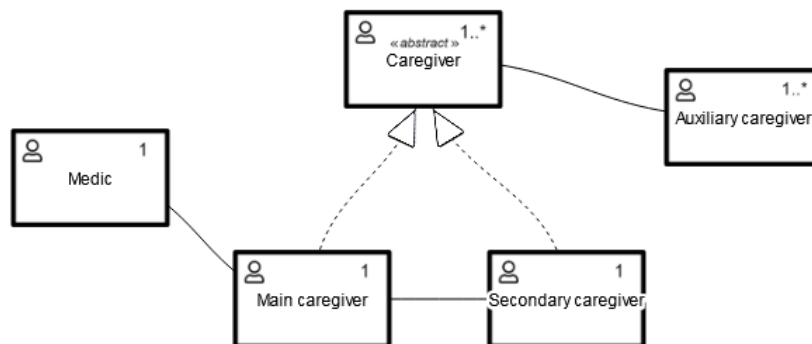


Figure 7.8. Role-interaction model (developer view) specified by the engineers.

Similar to the previous case, after explaining the model and the system prototype to the stakeholder, semi-structured interviews were conducted to the engineers and the stakeholder. The instruments to interview the stakeholder and the engineers are detailed in Appendix III and Appendix IV respectively.

First, using the corresponding instrument, the stakeholder was asked to identify actors in any role in the previously described role-interaction model. Moreover, we asked for the interaction services required by participants in the caregiving process. We then asked her if the interaction scenario specification was able to represent the family interaction needs. Additionally, we asked her if, according to her perception, the engineers understood these needs.

All questions shown in Appendix III were answered in a correct/positive manner by the stakeholder, except items 1 (Do you consider that your husband is represented in some role within the diagram?) and 2 (Do you consider that your children are represented in some role within the diagram?). Particularly, the stakeholder answered for item 1:

“Yes, he is represented in the role: auxiliary caregiver” (S2).

Whereas for item 2, she answered:

“Yes, they are represented by the role: secondary caregiver” (S2).

However, according to the engineer's explanation, she understood the roles in an inverted manner. That fact could be due to the role names chosen by the engineers (i.e., “auxiliary caregiver” and “secondary caregiver”), which possibly do not give her an intuitive insight to the actual meaning. Despite this fact, she finally considered engineers understood the communication needs of her family.

Regarding the engineers, all questions shown in Appendix IV were answered in a correct/positive manner by them. The application requirements were “very clear” to one engineer, whereas they were only “clear” to the other one. Additionally, an engineer suggested the following:

“I suggest not showing the services' table located to the right of the prototypes' window to the stakeholder - that could confuse her” (E3).

That suggestion is reasonable. Although no stakeholder's confusion about that was noticed, probably illustrating to her more interfaces than necessary may not be the better option.

Finally, all the activities established at the beginning of Section 7.2 were successfully performed by the participants of this replica. However, a mistake was detected:

- The stakeholder understood the meaning of a couple of roles in an inverted manner. That fact could be because the chosen role names (i.e., “auxiliary caregiver” and “secondary caregiver”) might not give it an intuitive insight of the actual meaning.

7.2.3. Replica III: Elderly with Mobility Limitations

The family caregiving process involved an elderly male with mobility limitations due to age. The representative stakeholder was a daughter of the elderly, who is a secondary participant in the caregiving process. Her sister is the main caregiver, and her daughters contribute to the caregiving process. The stakeholder recognized that although she wants to be the main caregiver, her activities do not allow her. Despite this, she aids periodically in such a process.

After the requirements elicitation, the engineers built the role-interaction model shown in Figure 7.9. This model was explained to the stakeholder, and particularly the process to be conducted when using the new system. The interfaces of the generated prototype by CIMoL Modeler were introduced to the stakeholder, to show how the application would support the interactions among the family members. The developer view of this model is shown in Figure 7.10.

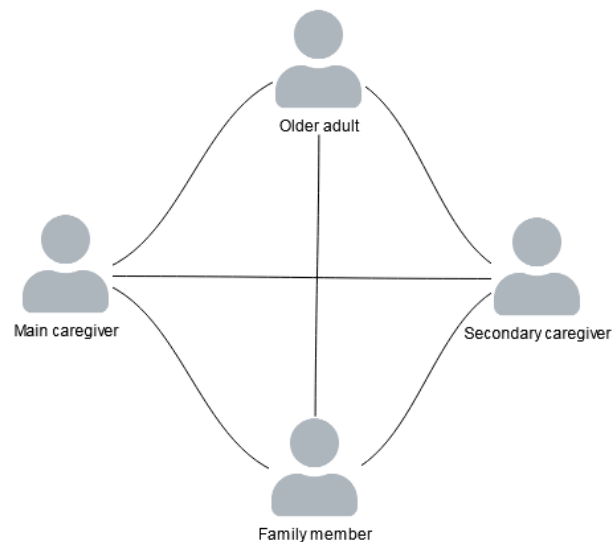


Figure 7.9. Role-interaction model (stakeholder view) specified by the engineers.

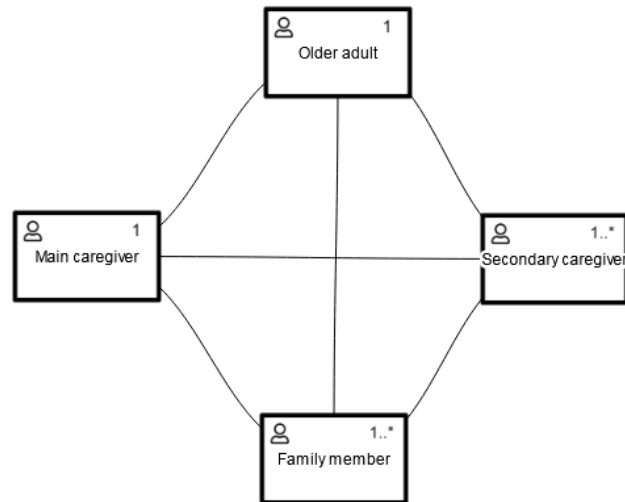


Figure 7.10. Role-interaction model (developer view) specified by the engineers.

Then, individual semi-structured interviews were performed to the engineers and the stakeholder. The instruments used in the interviews are detailed in Appendix V and Appendix VI respectively. The dynamic and the questions of these interviews were the same as the other replicas. All questions included in Appendix V were answered in a correct/positive manner by the stakeholder. However, there was a confusion with item 6. In this item, the stakeholder understood correctly that the medic was not represented in the diagram, but erroneously understood that communication with him could be allowed in the future. Specifically, she answered:

“I understood that the medic is not represented in the diagram, but I thought he could afterwards be added to the prototype as a member of the process” (S3).

We explained to her that adding afterwards the medic to the prototype was not possible. Despite this, she felt that engineers understood the communication needs of her family.

Regarding the engineers, all questions of the instrument shown in Appendix VI were answered in a correct/positive manner by them, also with exception of item 6. Some differences were noted in their answers; one of them answered:

“The medic was not considered in the diagram, but maybe we should consider him because the family could need him in the future” (E5).

Whereas the other engineer answered:

“We thought of adding the medic to the diagram. However, it could have a higher level of abstraction for the stakeholder and we finally decided not to include him” (E6).

Although such a fact could indicate a remarkable omission in this process analysis, we consider it is understandable, since the engineers are not specialists in the elderly-caregiving process. Furthermore, the application requirements were “very clear” to both engineers.

Finally, with the exception of activity 3 (since is-a relationships were not used in the modeling activity) and partially activity 5 (since abstract nodes were not required), all remaining activities stated at the beginning of the Section 7.2 were addressed by the participants of this replica. No errors were identified in the resulting models. However, the need of counting on a medic in the process (as an end-user) requires more discussion between stakeholders and engineers.

7.2.4. Replica IV: Elderly with Mobility Limitations

This caregiving process involved an elderly female with mobility limitations due to age. The representative stakeholder was a niece of the elderly, which is one of the responsible caregivers in the process. Furthermore, her husband, her mother-in-law, and her neighbor, also contribute to the caregiving process.

After the requirements elicitation, the engineers built the role-interaction model shown in Figure 7.11. Similar to the previous replicas, this model was used to explain to the stakeholder how the process would be conducted with the new system. Moreover, system prototypes were shown to the stakeholder to determine if the application deals with the interaction needs of the family members. The developer view of this model is illustrated in Figure 7.12.

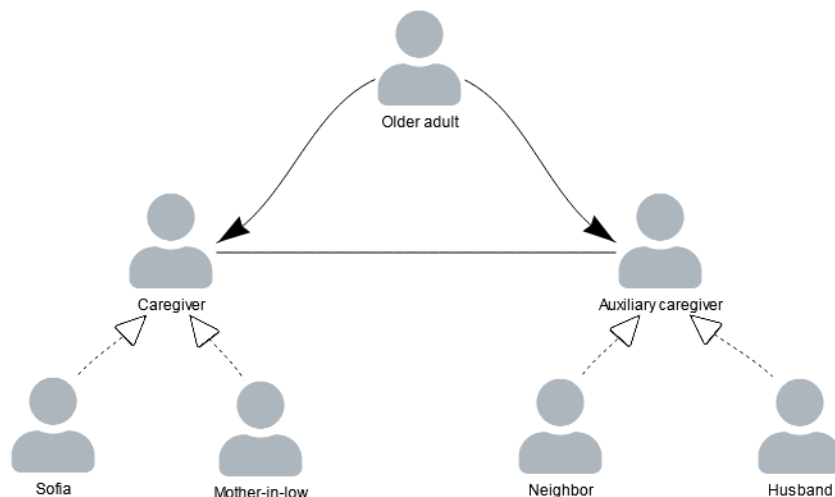


Figure 7.11. Role-interaction model (stakeholder view) specified by the engineers.

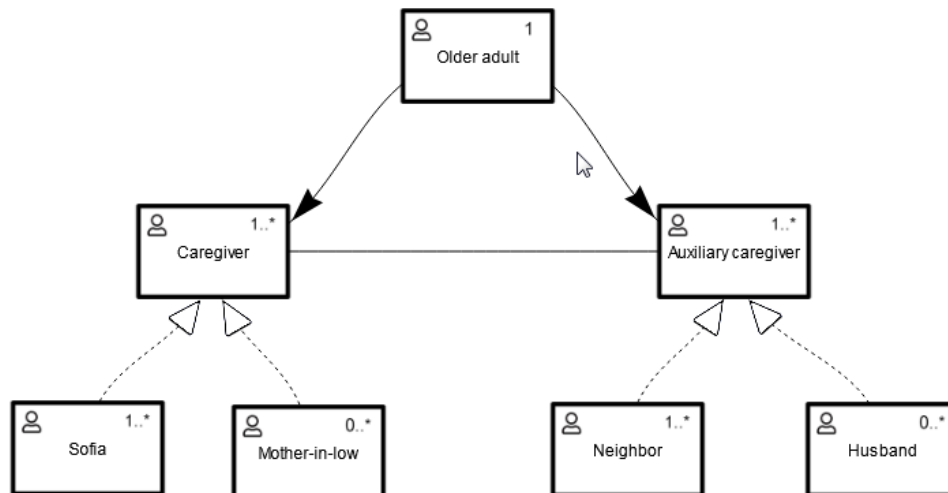


Figure 7.12. Role-interaction model (developer view) specified by the engineers.

In this replica, one particular situation took place. As we can observe in the models (Fig. 7.11 and 7.12), the engineers decided to assign a proper name (e.g., “Sofia”) or relatedness with the elderly (e.g., “Mother-in-law”) to the roles participating in the caregiving process. Although the explanation of the model and the prototypes were normally presented to the stakeholder, that activity had some different impacts in the interviews, mainly to engineers. The interviews to the stakeholder and engineers are detailed in Appendix VII and Appendix VIII respectively.

Analogously to the previous replicas, the stakeholder was asked to identify actors in any role and interaction services required by them. Then, we asked for the capability of the model’s representation to capture the family needs, and also to ease the building of a shared understanding of these needs.

All questions of Appendix VII were answered in a correct/positive manner by the stakeholder. For instance, for item 2 (do you consider that your mother-in-law is represented in some role within the diagram?) she answered:

“Yes, she is represented with the box named “mother-in-law”, which is a caregiver in the diagram” (S4).

As shown, she understood that the *mother-in-law* corresponds to a *caregiver* role. We noticed that this particular representation (i.e., to set the names or relatedness of actors playing the roles) could influence stakeholders to a positive understanding of the interaction scenarios. Similar answers were obtained for the remaining items. Finally, the stakeholder considered that engineers understood the interaction needs of the family caregivers.

Concerning the engineers, they answered all questions of Appendix VIII in a correct/positive manner. However, a discrepancy between the engineers became evident in an interview. In particular, an engineer indicated:

“I considered that all actors’ names or aliases (e.g., Sofia, mother-in-law, etc.) should not appear in the model because they are not roles. However, my colleague disagreed with that. He considered that by setting these names, the stakeholder could better understand the model, but I think it is not correct” (E8).

At the end, the first engineer yielded to not causing a problem, and declared:

“I told him (to the other engineer), we will do this part of the diagram as he suggests, no problem” (E8).

As explained previously, some real names or relatedness of the caregiving participants were used in the model, which caused a discrepancy between the engineers. This situation, rather than being a problem, makes us reflect about the importance of counting on close-to-reality labels in the model, to ease the understanding of the models by the stakeholders. This could help minimize errors in the communication requirements’ specification. Furthermore, the application requirements were considered “very clear” to one engineer (E7), whereas they were “clear” to the other one (E8).

The activities established at the beginning of Section 7.2 were successfully addressed by the participants of this replica. However, the following mistake was identified:

- Although the engineers were explained how to model an interaction graph using the CIMoL modeling tool, the task was not performed as defined. Four "instances" of the roles were identified, instead of the proper roles required to model the scenario using our approach. However, as mentioned previously, such an obvious and premeditated mistake related to the use of roles made us reflect on the need to have labels (close-to-reality) in the model, as a way to facilitate the understanding of the models by the stakeholders.

7.3. Discussion

The case study has positive results. A shared understanding about the interaction scenarios was achieved in the four replicas, according to the observed and analyzed situations. The model simplicity, together with the automatic prototyping, were helpful at the moment of explaining to stakeholders the interaction between roles. This also helped visualize the application (and its interaction services) required to support a family during the caregiving process.

Although most engineers were able to model the roles and the interactions correctly, and evidenced understanding of the stakeholders’ needs, the study showed some limitations

of the language. The main shortcoming is the need of adding components closer to stakeholders' language. The discrepancy between the engineers that happened during the last replica opens the opportunity to count on elements in the model to complement the stakeholders' understanding; e.g., working with "instances" analogously to UML. Thus, errors in the communication requirements' specification could be minimized.

Considering the research questions defined for this case study, we next analyze the answers gathered in the four replicas.

Regarding RQ1 (Are developers, jointly with stakeholders, able to specify the computer-based interaction and awareness services required by the participants in the informal elderly caregiving process?), we found the results of the study as positive. Particularly, the developers were able to identify roles and specify the interaction scenarios based on the information obtained from the first elicitation meeting. Then, developers together with stakeholders validated the model and specified the interaction and awareness services to be embedded in the mobile system supporting the caregiving process. The stakeholders did not experience significant difficulties in understanding the notion of roles and interaction between roles. According to the participants, the models and prototypes generated by CIMoL Modeler during each replica were understandable for them. Moreover, the models and prototypes represented properly the roles, and also the interactions among them, which are required to conduct the caregiving process.

Regarding RQ2 (Are developers and stakeholders able to achieve a shared understanding on these computer-based interaction and awareness services?), the results show a favorable answer to this question. All stakeholders considered that the language helped explain their communication needs among participants in their caregiving process, and the engineers understood such needs and vice versa. All engineers considered the use of CIMoL, through CIMoL Modeler, as actually contributing to achieve a shared understanding about the interaction scenario.

Summarizing, and considering only the people-driven collaborative processes addressed in the case study, the obtained results are aligned with the stated hypotheses. On the one hand, CIMoL allows specifying the role-based interaction scenarios (i.e., models), that involve the participants in the elderly caregiving process, in a comprehensible way (H1). Moreover, CIMoL Modeler allows refining these models and automatically deriving prototypes from these models. On the other hand, the visual models, along with prototypes generated from them, allow software engineers and stakeholders to reach a shared understanding on the interaction services and awareness required to support the PDCP (H2); in this case, the elderly caregiving process.

7.4. Threats to Validity

Internal validity. We acknowledge that several factors that were not explicitly requested in the study may have influenced it. Among these factors we can mention the following: 1) the selection of participants, given that such a process was performed through a personal invitation to people that meet the selection criteria (see section 7.1.3); 2) educational and cultural aspects of the caregivers that could have influenced their willingness for adopting technology to support the caregiving process; and 3) the participants' motivation may have affected the engagement of the people during the study.

External validity. We are aware that the results may be tied to the context where we conducted this study. Also, the participation of one author as a guide of the interviews may have biased the obtained results. However, such participation came down to reading each item of the questionnaire and clarifying the meaning of the items if required. We also are aware that the small number of participants poses a threat to the generalization of the results. For these reasons, we consider that additional studies in other contexts need to be conducted to better understand the factors that may influence the results obtained in this case study.

Reliability. To mitigate this threat to validity, the protocol used in the case study was designed following the guidelines proposed in (Runeson & Höst, 2008). Such a protocol was then reviewed by two researchers experienced in empirical software engineering. These researchers also helped us analyze and refine the interview questions. Additionally, the empirical data obtained from the case study was reviewed and discussed with these researchers, spending enough time in analyzing the results of each replica and trying to contradict the work hypotheses.

7.5. Summary

This chapter presents the case study that was designed and executed to evaluate the stated hypotheses. This evaluation was done by determining the capabilities of CIMoL to specify the computer-based interaction and awareness services required by the participants in an informal elderly caregiving process. The case study also helped us infer how much CIMoL contributes to achieve a shared understanding among engineers and stakeholders about these services.

Real software engineers and stakeholders (in this case, elderly caregivers) were involved in the study. The stakeholders were representatives of families located in Yucatan, Mexico, who have been performing informal elderly caregiving during a long time period. Something similar happens with the software engineers that participated in the case study.

The results indicate that CIMoL, together with the automatic prototyping capability provided by CIMoL Modeler, contribute to specify in a suitable manner the interaction and awareness services to be embedded in the mobile collaborative application that is being conceived; i.e., the interaction and awareness requirements of the system. Moreover, these tools also help software engineers and stakeholders achieve a shared understanding of the services. Regardless of these positive results, some elements of CIMoL were identified as potential sources of misunderstandings of interaction scenarios specification, due to the differences between the language used by stakeholders and engineers to explain and represent the PDCPs. In particular, we identified that some engineers would like to use components closer to stakeholders' language, to minimize the difference of meaning in the models' representation. This aspect will be explored as part of the future work.

Chapter 8. Conclusions and Future Work

This chapter presents the conclusions of the thesis. We first analyze the achievement of the research goals stated in Chapter 1. Then we present the theoretical and practical contributions of this research. Additionally, we discuss the limitations and future directions of this work.

8.1. Goals Achievement

The main research goal of this thesis was *to develop a machine-processable visual modeling language that allows developers and stakeholders to specify, analyze and validate interaction and awareness services to be embedded in mobile collaborative applications that support people-driven collaborative processes*. Thus, it complements current requirements engineering techniques for the development of systems supporting these processes.

The main goal was achieved through the definition of two components: The *Computer-Mediated Interactions Modeling Language* (CIMoL) and the CIMoL Modeler. The first one is a visual modeling language that provides the constructs to help software engineers and stakeholders represent the structure of PDCPs; particularly, processes that can be large and complex in terms of scenarios (i.e., phases and work ambits). The language also allows specifying the participants and interactions requirements for each scenario (Specific Goal 1).

The capabilities of this language to address the modeling challenges (presented in Section 1.3), were evaluated through an expert study and compared to other languages reported in the literature. The results (reported in Chapter 5) suggest that CIMoL provides better support to deal with these challenges than the other languages.

The second component, i.e., CIMoL Modeler, is a software modeling tool that helps software engineers and stakeholders to visually model both the structure of a PDCP and its interaction scenarios using CIMoL (Specific Goal 2). This tool uses the models of interaction scenarios as input to automatically generate a prototype of the interfaces of the collaborative system that provides interaction support to the participants in the PDCP process (*Specific Goal 3*).

The usefulness of this tool was evaluated through a case study that involved four replicas, whose goal was to model the roles and the interactions between them for a mobile application to support the informal elderly caregiving process. This case study included the participation of real caregivers (as stakeholders) and software engineers (as solution providers), who worked together to reach an agreement on the interaction and awareness services required by the supporting system. The results of this case study (see Chapter 7) indicate that the use of CIMoL Modeler can help stakeholders and

developers to both reach an agreement on these interaction requirements and build a shared understanding of these services. Gathering stronger evidence by means of further validation is proposed as future work.

8.2. Contributions to Theory and Practice

This section describes the theoretical and practical contributions of this research to the field of people-driven collaborative processes. As part of the theoretical contributions we can mention the following ones:

The analysis of capabilities and limitations of existing languages and notations for modeling PDCPs. This theoretical analysis provides the understanding of visual languages and notations with capabilities to model interactions between roles involved in people-driven collaborative processes. Chapter 3 presents this analysis, based on the assessment of modeling languages' experts; most of them were the authors of such proposals. Therefore, the reported capabilities and limitations provide an informed perspective of the state-of-the-art in visual languages and notations to model interaction scenarios. To the best of our knowledge, there is no other research work analyzing the state-of-the-art in this study domain.

The Computer-Mediated Interactions Modeling Language. The Computer-Mediated Interactions Modeling Language (CIMoL) was defined to model interactions among roles in scenarios involved in people-driven collaborative processes. CIMoL, which is presented in Chapter 4, was conceived considering the limitations of the existing modeling languages, and was designed using the principles of visual languages' design from Moody (2009). The expert study presented in Chapter 5 shows a strong indication that CIMoL could address the design challenges of PDCPs in a better way than other languages and notations reported in the literature.

As part of the practical contributions, this research work provides support to the activities performed by practitioners; particularly, those conducted by software engineers and stakeholders during the requirements engineering process at project pre-selling or during its execution. Such support is mainly provided by CIMoL Modeler (presented in Chapter 6) through the following services:

Modeling of interaction scenarios and setting of the interaction services. This visual modeling tool allows representing the structure of PDCPs and their interaction scenarios. Moreover, it lets the users set the interactions properties at design time of an interaction scenario. As result, it creates a computer-processable model that adheres to the metamodel described in Section 4.1. This service uses the two visual dialects defined by CIMoL, and also a list of predefined interaction and awareness services, to help stakeholders and engineers achieve an agreement on the interaction requirements to be considered by the mobile application supporting the PDCP.

Automatic generation of system prototypes. This service uses an interaction scenario model as input, to automatically generate the interfaces of a prototype that support the interactions specified in such a model. In that, the interaction and awareness services set in the model are translated into their visual representations in the prototype. Such a correspondence (*service, visual representation*) can help improve the stakeholders' understanding of the software services specified in the interaction models. Finally, the prototypes, jointly with the visual models, can help stakeholders and engineers build a shared understanding about the interaction and awareness services to be embedded in the application being developed.

8.3. Limitations

The limitations of this research work are listed below.

- The scope of this thesis is limited to modeling roles and the interaction between them, in interaction scenarios involved in people-driven collaborative processes. Modeling other aspects of these processes (e.g., the business logic) will require the use of other modeling languages.
- The use of CIMoL would be less beneficial than informal modeling (or ad-hoc modeling) when the number of roles participating in a process is very small or the process has no clearly defined roles. For instance, when the process has one or two simple roles, or when in the process there is a people community in which the process understanding depends on particular persons without predefined roles. In such a situation, modeling the process in an ad-hoc manner could be more appropriate and using CIMoL could give us no advantage.
- Similar to any other visual modeling language (e.g., UML), CIMoL has limitations when the number of entities to model is large. That is, when the number of roles and interactions for a particular scenario overcomes the twenty, the models start to increasingly lose understandability.
- Although the obtained results support the stated hypotheses, the practical validation was carried out in only one application domain, with a limited number of cases. This does not allow us to generalize the conclusions on the capabilities of the modeling language. Therefore, more evaluations, including other PDCPs, are required.
- Although we have positive indicators concerning the understandability of the models by stakeholders, this requires further research considering diverse application domains and more stakeholders.

- The study results have demonstrated the applicability of the modeling language and the supporting tool in a real scenario. However, it is still pending to analyze the voluntary adoption of these technologies by practitioners in industrial settings.
- The interactions setting and the prototype generator are limited by the pre-established interaction and awareness services currently considered in CIMoL Modeler. This set of services is not complete, but it is comprehensive (Canché et al., 2022a) and extensible. It means that if we want to use a new service we must modify the code of the interactions setting and the prototype generator.

8.4. Future Work

The limitations discussed in the previous section leads us to indicate a number of future research directions, which are explained below.

1. Extend CIMoL to allow engineers to refer to roles in terms of the stakeholders' understanding. During the experiments, we saw that stakeholders tend to refer to actors (i.e., instances, actual people participating in the interaction scenarios) and not to roles (i.e., the type of actors that participate). Extending CIMoL to allow engineers to also capture role instances along with roles themselves, will ease building a shared understanding between engineers and stakeholders.
2. Perform additional validations to CIMoL and CIMoL Modeler, involving PDCPs from several application domains and a significant number of stakeholders for each process. The understandability of the models by stakeholders is a priority aspect to be validated. This allows us to visualize the capabilities of these components in a more comprehensive way, and also to reduce the threats to the generalization of the language. These validations also will contribute to determine strengths and weaknesses of the proposal to be adopted by the software industry.
3. Extend the CIMoL Modeler to create more complete system prototypes, thus improving the capability of stakeholders and engineers to reach an agreement and shared understanding on the product scope. This extension should address two main aspects. The first one is to allow performing the interactions validations and prototypes generation based on a set of interaction scenarios, instead of a single model. This will contribute to create systems prototypes more comprehensive in terms of interaction support. The second aspect is to allow integrating the part of the business logic of the application (particularly that available through the user interface) into the system models. This will let the system create mockups with a functionality closer to the real application, and thus, easing the achievement of agreements between stakeholders and software engineers.

4. Design mechanisms to import and export interaction models between CIMoL and other modeling languages. This will contribute to reusing design solutions that support PDCPs; reducing thus the effort required to conceive and extend them.

Glossary

Next we briefly define the main concepts used in this thesis work:

Client organization. It is also known as the contractor or customer. Typically, it represents the organization that requests a project proposal or development.

Human-machine unit. A mobile worker that uses a collaborative application to participate in a PDCP. Typically, these HMUs perform loosely-coupled work, i.e., the HMUs work autonomously most of the time and carry out sporadic on-demand collaboration processes. Once finished the collaboration activity, these units return to autonomous work.

Product scope. The product scope is a definition that includes the do's and don'ts of the product being developed. It represents the prospect of the project for stakeholders and the provider (Hassan & Asghar, 2021; Taherdoost & Abolfazl, 2018).

Product scoping. It is also known as product scope definition in which stakeholders and software engineers participate. It is an early and iterative requirements engineering activity (Vera et al., 2021b) that involves elicitation, specification and validation of the scope of the product to be developed.

Software engineers. A *software engineer* is a member of the development team, who can be involved in the specification, design, construction, deployment, evolution, and maintenance of software systems. Software engineers can play several roles; for instance, they can be team leaders, requirements engineers, architects, designers, programmers and testers (Glinz & Fricker, 2015).

Software provider. Typically, this is the software organization in charge of developing the software system; this is also named the supplier or developer.

Stakeholders. A *stakeholder* is a person of the client organization that has a direct or indirect influence on system requirements. Typical stakeholders are clients, users, and managers.

Bibliography

- Aalst, W. M. P. van der, Weske, M., & Grünbauer, D. (2005). Case handling: A new paradigm for business process support. *Data & Knowledge Engineering*, 53(2), 129–162. <https://doi.org/10.1016/j.datak.2004.07.003>
- Aldunate, R., Ochoa, S. F., Peña-Mora, F., & Nussbaum, M. (2006). Robust Mobile Ad Hoc Space for Collaboration to Support Disaster Relief Efforts Involving Critical Physical Infrastructure. *Journal of Computing in Civil Engineering*, 20(1), 13–27. [https://doi.org/10.1061/\(ASCE\)0887-3801\(2006\)20:1\(13\)](https://doi.org/10.1061/(ASCE)0887-3801(2006)20:1(13))
- Allah Bukhsh, Z., van Sinderen, M., Sikkel, K., & Quartel, D. (2019). How to Manage and Model Unstructured Business Processes: A Proposed List of Representational Requirements. In M. S. Obaidat & E. Cabello (Eds.), *E-Business and Telecommunications* (pp. 81–103). Springer International Publishing. https://doi.org/10.1007/978-3-030-11039-0_5
- Antonaya, S. L., & Santos, C. B. (2010). Towards a framework for the development of CSCW systems. In *Cooperative design, visualization, and engineering* (pp. 117–120). Springer.
- Antunes, P., Xiao, L., & Pino, J. A. (2014). Assessing the impact of educational differences in HCI design practice. *International Journal of Technology and Design Education*, 24(3), 317–335.
- Baldwin, C. Y., & Clark, K. B. (2000). *Design rules: The power of modularity* (Vol. 1). MIT press.
- Barros, C., Fernández, M. B., & Herrera, M. S. (2014). *Notions that people over 45 years old have about family and the internal interactions between family members [in Spanish]. Psicoperspectivas 13 (1): 121–130.*

- Baxter, G., & Sommerville, I. (2011). Socio-technical systems: From design methods to systems engineering. *Interacting with Computers*, 23(1), 4–17.
- Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for agile software development*. <http://www.agilemanifesto.org/> [Accessed 25 August 2021]
- Ben Hassen, M., Turki, M., & Gargouri, F. (2017). Towards Extending Business Process Modeling Formalisms with Information and Knowledge Dimensions. In S. Benferhat, K. Tabia, & M. Ali (Eds.), *Advances in Artificial Intelligence: From Theory to Practice* (pp. 407–425). Springer International Publishing.
- Ben Hassen, M., Turki, M., & Gargouri, F. (2019). A Multi-criteria Evaluation Approach for Selecting a Sensitive Business Process Modeling Language for Knowledge Management. *J. Data Semant.*, 8(3), 157–202. <https://doi.org/10.1007/s13740-019-00103-5>
- Berander, P., & Andrews, A. (2005). Requirements Prioritization. In A. Aurum & C. Wohlin (Eds.), *Engineering and Managing Software Requirements* (pp. 69–94). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-28244-0_4
- Bittner, E. A. C., & Leimeister, J. M. (2013). Why Shared Understanding Matters – Engineering a Collaboration Process for Shared Understanding to Improve Collaboration Effectiveness in Heterogeneous Teams. *2013 46th Hawaii International Conference on System Sciences*, 106–114. <https://doi.org/10.1109/HICSS.2013.608>
- Boehm, B. W. (2002). Software Engineering Economics. In M. Broy & E. Denert (Eds.), *Software Pioneers: Contributions to Software Engineering* (pp. 641–686). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-59412-0_38

- Britton, C., & Jones, S. (1999). The Untrained Eye: How Languages for Software Specification Support Understanding in Untrained Users. *Human-Computer Interaction*, 14(1-2), 191-244. <https://doi.org/10.1080/07370024.1999.9667269>
- Bubaš, G. (2001). Computer mediated communication theories and phenomena: Factors that influence collaboration over the Internet. *3rd CARNet Users Conference. Zagreb, September, 2001*, 24-26.
- Canché, M., & Ochoa, S. F. (2018). Modeling Computer-Mediated User Interactions in Ubiquitous Collaborative Systems. In J. Bravo & O. Baños (Eds.), *12th International Conference on Ubiquitous Computing and Ambient Intelligence, UCAMI 2018, Punta Cana, Dominican Republic, December 4-7, 2018* (Vol. 2, Issue 19, p. 1250). MDPI. <https://doi.org/10.3390/proceedings2191250>
- Canché, M., & Ochoa, S. F. (2019). A Survey of Development Strategies for Collaborative Systems. In W. Shen, H. Paredes, J. Luo, & J.-P. A. Barthès (Eds.), *23rd IEEE International Conference on Computer Supported Cooperative Work in Design, CSCWD 2019, Porto, Portugal, May 6-8, 2019* (pp. 261-266). IEEE. <https://doi.org/10.1109/CSCWD.2019.8791934>
- Canché, M., Ochoa, S. F., & Perovich, D. (2021). *CIMoL: A Language for Modeling Interactions in People-driven Collaborative Processes* (Technical Report TR/DCC-2021-3). Computer Science Department. University of Chile. https://www.dcc.uchile.cl/TR/2021/TR_DCC-20211115-003.pdf
- Canché, M., Ochoa, S. F., & Perovich, D. (2022a). Identifying Interaction and Awareness Services in Mobile Collaborative Applications. *24th International Conference on Human-Computer Interaction, HCI International 2022, Gothenburg, Sweden, 26 June - 1 July, 2022*, 3-16.
- Canché, M., Ochoa, S. F., & Perovich, D. (2022b). Modelling Interactions Happening in

- People Driven Collaborative Processes. In G. Fortino, D. Kaber, A. Nürnberger, & D. Mendonca (Eds.), *Handbook of Human-Machine Systems*. IEEE Press/Wiley. In press.
- Canché, M., Ochoa, S. F., & Perovich, D. (2022c). Understanding the Suitability of Modeling Languages and Notations to Represent Computer-Mediated Interaction Scenarios. *Lecture Notes in Networks and Systems*. https://doi.org/10.1007/978-3-030-96293-7_57
- Canché, M., Ochoa, S. F., Perovich, D., & Gutierrez, F. J. (2019). Analysis of notations for modeling user interaction scenarios in ubiquitous collaborative systems. *Journal of Ambient Intelligence and Humanized Computing*. <https://doi.org/10.1007/s12652-019-01578-7>
- Canché, M., & Pino, J. A. (2021). Requirements Elicitation for Collaborative Systems: A Systematic Review. *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 297–304.
- Cardoso, E., Labunets, K., Dalpiaz, F., Mylopoulos, J., & Giorgini, P. (2016). Modeling Structured and Unstructured Processes: An Empirical Evaluation. In I. Comyn-Wattiau, K. Tanaka, I.-Y. Song, S. Yamamoto, & M. Saeki (Eds.), *Conceptual Modeling* (pp. 347–361). Springer International Publishing.
- Carrizo, D., Dieste, O., & Juristo, N. (2014). Systematizing requirements elicitation technique selection. *Information and Software Technology*, 56(6), 644–669.
- Cass, A. G., Lerner, B. S., Sutton, S. M., Jr., McCall, E. K., Wise, A., & Osterweil, L. J. (2000). Little-JIL/Juliette: A Process Definition Language and Interpreter. *Proceedings of the 22Nd International Conference on Software Engineering*, 754–757. <https://doi.org/10.1145/337180.337623>
- Cheng, P. C.-H. (2004). Why Diagrams Are (Sometimes) Six Times Easier than Words:

- Benefits beyond Locational Indexing. In A. F. Blackwell, K. Marriott, & A. Shimojima (Eds.), *Diagrammatic Representation and Inference* (pp. 242–254). Springer Berlin Heidelberg.
- Churchill, E. F., & Wakeford, N. (2002). Framing mobile collaborations and mobile technologies. In *Wireless world* (pp. 154–179). Springer.
- Clarke, E., Grumberg, O., Kroening, D., Peled, D., & Veith, H. (2018). *Model Checking, second edition*. MIT Press.
- Clarke, E., McMillan, K., Campos, S., & Hartonas-Garmhausen, V. (1996). Symbolic model checking. *International Conference on Computer Aided Verification*, 419–422.
- Collazos, C. A., Gutiérrez, F. L., Gallardo, J., Ortega, M., Fardoun, H. M., & Molina, A. I. (2019). Descriptive theory of awareness for groupware development. *Journal of Ambient Intelligence and Humanized Computing*, 10(12), 4789–4818.
- Comi, A., & Eppler, M. J. (2011). Visual representations as carriers and symbols of organizational knowledge. *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, 1–7.
- Conboy, K., & Fitzgerald, B. (2010). Method and Developer Characteristics for Effective Agile Method Tailoring: A Study of XP Expert Opinion. *ACM Trans. Softw. Eng. Methodol.*, 20(1), 1–30. <https://doi.org/10.1145/1767751.1767753>
- Coughlan, J., & Macredie, R. D. (2002). Effective Communication in Requirements Elicitation: A Comparison of Methodologies. *Requirements Engineering*, 7(2), 47–60. <https://doi.org/10.1007/s007660200004>
- Cramton, C. D. (2001). The Mutual Knowledge Problem and Its Consequences for Dispersed Collaboration. *Organization Science*, 12(3), 346–371. <https://doi.org/10.1287/orsc.12.3.346.10098>

- Cruz, W. M., Isotani, S., Carriço, L., & Guerreiro, T. (2016). Interface to Support Caregivers in Daily Record and Information Visualization of Patients with Dementia. *Proceedings of the 15th Brazilian Symposium on Human Factors in Computing Systems*, 13:1-13:10. <https://doi.org/10.1145/3033701.3033714>
- Dalpiaz, F., Franch, X., & Horkoff, J. (2016). Istar 2.0 language guide. *ArXiv Preprint ArXiv:1605.07767*.
- Dangarska, Z., Figl, K., & Mendling, J. (2016). An Explorative Analysis of the Notational Characteristics of the Decision Model and Notation (DMN). *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)*, 1–9. <https://doi.org/10.1109/EDOCW.2016.7584345>
- Darch, P., Carusi, A., & Jirotko, M. (2009). Shared understanding of end-users' requirements in e-Science projects. *2009 5th IEEE International Conference on E-Science Workshops*, 125–128. <https://doi.org/10.1109/ESCIW.2009.5407963>
- Deng, S., Huang, L., Wu, H., Tan, W., Taheri, J., Zomaya, A. Y., & Wu, Z. (2016). Toward Mobile Service Computing: Opportunities and Challenges. *IEEE Cloud Computing*, 3(4), 32–41. <https://doi.org/10.1109/MCC.2016.92>
- Di Ciccio, C., Marrella, A., & Russo, A. (2015). Knowledge-Intensive Processes: Characteristics, Requirements and Analysis of Contemporary Approaches. *Journal on Data Semantics*, 4(1), 29–57. <https://doi.org/10.1007/s13740-014-0038-4>
- Dirix, M., Le Pallec, X., & Muller, A. (2014). Software Support Requirements for Awareness in Collaborative Modeling. In R. Meersman, H. Panetto, T. Dillon, M. Missikoff, L. Liu, O. Pastor, A. Cuzzocrea, & T. Sellis (Eds.), *On the Move to Meaningful Internet Systems: OTM 2014 Conferences* (pp. 382–399). Springer Berlin Heidelberg.

- Dorn, C., & Dustdar, S. (2011). Supporting Dynamic, People-Driven Processes through Self-learning of Message Flows. In H. Mouratidis & C. Rolland (Eds.), *Advanced Information Systems Engineering* (pp. 657–671). Springer Berlin Heidelberg.
- Dorn, C., Dustdar, S., & Osterweil, L. J. (2014). Specifying Flexible Human Behavior in Interaction-Intensive Process Environments. In S. Sadiq, P. Soffer, & H. Völzer (Eds.), *Business Process Management* (pp. 366–373). Springer International Publishing.
- Dorn, C., & Taylor, R. N. (2012). Architecture-Driven Modeling of Adaptive Collaboration Structures in Large-Scale Social Web Applications. In X. S. Wang, I. Cruz, A. Delis, & G. Huang (Eds.), *Web Information Systems Engineering—WISE 2012* (pp. 143–156). Springer Berlin Heidelberg.
- Dwyer, J. W., & Coward, R. T. (1991). A multivariate comparison of the involvement of adult sons versus daughters in the care of impaired parents. *Journal of Gerontology*, 46(5), S259–S269. <https://doi.org/10.1093/geronj/46.5.S259>
- Ehrlich, K. (1999). Designing groupware applications: A work-centered design approach. *Computer Supported Co-Operative Work*, 1, 28.
- Ellis, C. A., Gibbs, S. J., & Rein, G. (1991). Groupware: Some Issues and Experiences. *Commun. ACM*, 34(1), 39–58. <https://doi.org/10.1145/99977.99987>
- Ellis, C., & Wainer, J. (1994). A conceptual model of groupware. *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, 79–88. <https://doi.org/10.1145/192844.192878>
- Engels, G., Küster, J. M., Heckel, R., & Groenewegen, L. (2001). A methodology for specifying and analyzing consistency of object-oriented behavioral models. *ACM SIGSOFT Software Engineering Notes*, 26(5), 186–195.
- Eppler, M. J. (2006). A comparison between concept maps, mind maps, conceptual

- diagrams, and visual metaphors as complementary tools for knowledge construction and sharing. *Information Visualization*, 5(3), 202–210.
- Esbensen, A. J. (2010). Health conditions associated with aging and end of life of adults with Down syndrome. *International Review of Research in Mental Retardation*, 39, 107–126.
- Ewenstein, B., & Whyte, J. K. (2007). Visual representations as ‘artefacts of knowing.’ *Building Research & Information*, 35(1), 81–89.
- Fischer, G., & Herrmann, T. (2015). Meta-design: Transforming and enriching the design and use of socio-technical systems. In *Designing Socially Embedded Technologies in the Real-World* (pp. 79–109). Springer.
- Fisher, A., Nolan, M., Friedenthal, S., Loeffler, M., Sampson, M., Bajaj, M., VanZandt, L., Hovey, K., Palmer, J., & Hart, L. (2014). Model lifecycle management for MBSE. *INCOSE International Symposium*, 24(1), 207–229.
- Frank, U. (2013). Domain-specific modeling languages: Requirements analysis and design guidelines. In *Domain engineering* (pp. 133–157). Springer.
- Fuks, H., Raposo, A., Gerosa, M. A., Pimentel, M., Filippo, D., & Lucena, C. (2008). Inter- and intra-relationships between communication coordination and cooperation in the scope of the 3C Collaboration Model. *12th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2008)*, Xi’an, China, 16-18 April 2008, 1, 148–153. <https://doi.org/10.1109/CSCWD.2008.4536971>
- Gemino, A., & Wand, Y. (2005). Complexity and clarity in conceptual modeling: Comparison of mandatory and optional properties. *Data & Knowledge Engineering*, 55(3), 301–326. <https://doi.org/10.1016/j.datak.2004.12.009>
- Gerosa, M. A., Pimentel, M., Raposo, A. B., Fuks, H., & de Lucena, C. J. P. (2005).

- Towards an engineering approach for groupware development: Learning from the AulaNet LMS development. *Proceedings of the Ninth International Conference on Computer Supported Cooperative Work in Design (CSCWD 2005), Coventry, UK, 24-26 May 2005.*, 1, 329–333. <https://doi.org/10.1109/CSCWD.2005.194192>
- Glinz, M. (2014). *A Glossary of Requirements Engineering Terminology*. <https://www.ireb.org/en/cpre/cpre-glossary/>
- Glinz, M., & Fricker, S. A. (2015). On shared understanding in software engineering: An essay. *Computer Science - Research and Development*, 30(3), 363–376. <https://doi.org/10.1007/s00450-014-0256-x>
- Goldschmidt, G. (1997). Capturing indeterminism: Representation in the design problem space. *Design Studies*, 18(4), 441–455.
- Grünbacher, P., & Seyff, N. (2005). Requirements Negotiation. In A. Aurum & C. Wohlin (Eds.), *Engineering and Managing Software Requirements* (pp. 143–162). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-28244-0_7
- Guareis de farias, C. R., Ferreira Pires, L., & van Sinderen, M. (2000a). A component-based groupware development methodology. *Proceedings Fourth International Enterprise Distributed Objects Computing Conference. EDOC2000*, 204–213.
- Guareis de farias, C. R., Ferreira Pires, L., & van Sinderen, M. (2000b). A conceptual model for the development of CSCW systems. In R. Dieng-Kuntz, A. Giboin, L. Karsenty, & G. De Michelis (Eds.), *Designing cooperative systems: The use of theories and models: Proceedings of the 5th International Conference on the Design of Cooperative Systems (COOP'2000)* (pp. 189–204). IOS Press.
- Guarino, N., Oberle, D., & Staab, S. (2009). What Is an Ontology? In *Handbook on Ontologies* (pp. 1–17). https://doi.org/10.1007/978-3-540-92673-3_0

- Gutierrez, F. J., & Ochoa, S. F. (2017). It Takes at Least Two to Tango: Understanding the Cooperative Nature of Elderly Caregiving in Latin America. *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17), Portland, Oregon, USA, 25 February - 1 March 2017*, 1618–1630. <https://doi.org/10.1145/2998181.2998314>
- Gutierrez, F. J., Ochoa, S. F., & Vassileva, J. (2016). Identifying opportunities to support family caregiving in Chile. *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2112–2118. <https://doi.org/10.1145/2851581.2892386>
- Hassan, I. U., & Asghar, S. (2021). A Framework of Software Project Scope Definition Elements: An ISM-DEMATEL Approach. *IEEE Access*, 9, 26839–26870. <https://doi.org/10.1109/ACCESS.2021.3057099>
- Hawryszkiewicz, I. T. (2005). A Metamodel for Modeling Collaborative Systems. *Journal of Computer Information Systems*, 45(3), 63–72. <https://doi.org/10.1080/08874417.2005.11645844>
- Hawryszkiewicz, I. T. (2009). Modeling Complex Adaptive Systems. In J. Yang, A. Ginige, H. C. Mayr, & R.-D. Kutsche (Eds.), *Information Systems: Modeling, Development, and Integration* (pp. 458–468). Springer Berlin Heidelberg.
- Herskovic, V., Ochoa, S. F., & Pino, J. A. (2019). Identifying Groupware Requirements in People-Driven Mobile Collaborative Processes. *J. Univers. Comput. Sci.*, 25(8), 988–1017.
- Herskovic, V., Ochoa, S. F., Pino, J. A., Antunes, P., & Ormeño, E. (2013). Identifying the awareness mechanisms for mobile collaborative applications. *International Conference on Collaboration and Technology*, 241–256.
- Hickey, A., & Davis, A. (2002). The role of requirements elicitation techniques in

achieving software quality. *International Workshop on Requirements Engineering: Foundations for Software Quality (REFSQ)*.

Hoffmann, A., Bittner, E. A. C., & Leimeister, J. M. (2013). The Emergence of Mutual and Shared Understanding in the System Development Process. In J. Doerr & A. L. Opdahl (Eds.), *Requirements Engineering: Foundation for Software Quality* (pp. 174–189). Springer Berlin Heidelberg.

Holzmann, G. J. (1997). The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5), 279–295.

Huth, C., Erdmann, I., & Nastansky, L. (2001). GroupProcess: Using Process Knowledge from the Participative Design and Practical Operation of ad hoc Processes for the Design of Structured Workflows. *34th Annual Hawaii International Conference on System Sciences (HICSS-34), January 3-6, 2001, Maui, Hawaii, USA*. <https://doi.org/10.1109/HICSS.2001.927236>

Irani, P., Tingley, M., & Ware, C. (2001). Using perceptual syntax to enhance semantic content in diagrams. *IEEE Computer Graphics and Applications*, 21(5), 76–84.

Irani, P., & Ware, C. (2003). Diagramming information structures using 3D perceptual primitives. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 10(1), 1–19.

Kalyuga, S., Ayres, P., Chandler, P., & Sweller, J. (2003). The Expertise Reversal Effect. *Educational Psychologist*, 38(1), 23–31. https://doi.org/10.1207/S15326985EP3801_4

Kastenbergh, H., & Rensink, A. (2006). Model checking dynamic states in GROOVE. *International SPIN Workshop on Model Checking of Software*, 299–305.

Komal, B., Janjua, U. I., Anwar, F., Madni, T. M., Cheema, M. F., Malik, M. N., & Shahid, A. R. (2020). The Impact of Scope Creep on Project Success: An Empirical

- Investigation. *IEEE Access*, 8, 125755–125775.
<https://doi.org/10.1109/ACCESS.2020.3007098>
- Kumari, N., & Pillai, A. S. (2014). A study on project scope as a requirements elicitation issue. *2014 International Conference on Computing for Sustainable Global Development (INDIACom)*, 510–514.
<https://doi.org/10.1109/IndiaCom.2014.6828190>
- Lampa, I. L., Contessoto, A. G., Amorim, A. R., Zafalon, G. F. D., Valêncio, C. R., & Souza, R. C. G. (2017). Project Scope Management: A Strategy Oriented to the Requirements Engineering. *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017), Volume 2, Porto, Portugal, April 26-29, 2017*, 370–378. <https://doi.org/10.5220/0006318603700378>
- Larkin, J. H., & Simon, H. A. (1987). Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science*, 11(1), 65–100.
[https://doi.org/10.1016/S0364-0213\(87\)80026-5](https://doi.org/10.1016/S0364-0213(87)80026-5)
- Lauesen, S. (2002). *Software Requirements: Styles and Techniques*. Addison-Wesley Professional.
- Lee, C. P., & Paine, D. (2015). From The matrix to a model of coordinated action (MoCA) A conceptual framework of and for CSCW. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, 179–194.
- Li, H., Kraut, R. E., & Zhu, H. (2021). Technical Features of Asynchronous and Synchronous Community Platforms and their Effects on Community Cohesion: A Comparative Study of Forum-based and Chat-based Online Mental Health Communities. *Journal of Computer-Mediated Communication*, 26(6), 403–421.
<https://doi.org/10.1093/jcmc/zmab016>

- Masri, K., Parker, D., & Gemino, A. (2008). Using iconic graphics in entity-relationship diagrams: The impact on understanding. *Journal of Database Management (JDM)*, 19(3), 22–41.
- McCarthy, J. (1993). Notes on Formalizing Context. *Proceedings of the 13th International Joint Conference on Artificial Intelligence - Volume 1*, 555–560.
- Mendling, J. (2009). Empirical Studies in Process Model Verification. In K. Jensen & W. M. P. van der Aalst (Eds.), *Transactions on Petri Nets and Other Models of Concurrency II: Special Issue on Concurrency in Process-Aware Information Systems* (pp. 208–224). Springer Berlin Heidelberg.
https://doi.org/10.1007/978-3-642-00899-3_12
- Miranda, I., Araujo, R. M. de, & Borges, M. R. S. (2007). Discovering Group Communication Requirements. In F. Losavio, G. H. Travassos, V. Pelechano, I. Díaz, & A. Matteo (Eds.), *Memorias de la X Conferencia Iberoamericana de Software Engineering (CibSE 2007), Isla de Margarita, Venezuela, Mayo 7-11, 2007* (pp. 107–120).
- Mirza, M. N., Pourzolfaghar, Z., & Shahnazari, M. (2013). Significance of Scope in Project Success. *Procedia Technology*, 9, 722–729.
<https://doi.org/10.1016/j.protcy.2013.12.080>
- Monares, A., Ochoa, S. F., Herskovic, V., Santos, R. M., & Pino, J. A. (2014). Modeling interactions in human-centric wireless sensor networks. In J.-L. Hou, A. J. C. Trappey, C.-W. Wu, K.-H. Chang, C.-S. Liao, W. Shen, J.-P. A. Barthès, & J. Luo (Eds.), *Proceedings of the IEEE 18th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2014, Hsinchu, Taiwan, May 21-23, 2014* (pp. 661–666). IEEE. <https://doi.org/10.1109/CSCWD.2014.6846923>
- Moody, D. (2009). The “Physics” of Notations: Toward a Scientific Basis for Constructing

- Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering*, 35(6), 756–779. <https://doi.org/10.1109/TSE.2009.67>
- Mumford, E. (2006). The story of socio-technical design: Reflections on its successes, failures and potential. *Information Systems Journal*, 16(4), 317–342.
- Muram, F. ul, Tran, H., & Zdun, U. (2017). Systematic review of software behavioral model consistency checking. *ACM Computing Surveys (CSUR)*, 50(2), 1–39.
- National Institutes of Health. (2022). *Aging in Place: Growing Older at Home*. U.S. Department of Health & Human Services. <https://www.nia.nih.gov/health/aging-place-growing-older-home> [Accessed 19 September 2022]
- Netto, J. M., França, J. B. S., Baião, F. A., & Santoro, F. M. (2013). A notation for Knowledge-Intensive Processes. *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 190–195. <https://doi.org/10.1109/CSCWD.2013.6580961>
- Nolte, A., & Prilla, M. (2013). Anyone can use models: Potentials, requirements and support for non-expert model interaction. *International Journal of E-Collaboration (IJeC)*, 9(4), 45–60.
- Nuseibeh, B., & Easterbrook, S. (2000). Requirements Engineering: A Roadmap. *Proceedings of the Conference on The Future of Software Engineering (ICSE '00)*, New York, NY, USA, May 2000, 35–46. <https://doi.org/10.1145/336512.336523>
- Ochoa, S., Alarcon, R., & Guerrero, L. (2009). Understanding the Relationship between Requirements and Context Elements in Mobile Collaboration. In J. A. Jacko (Ed.), *Human-Computer Interaction. Ambient, Ubiquitous and Intelligent Interaction* (pp. 67–76). Springer Berlin Heidelberg.

- Ochoa, S. F., Bravo, G., Pino, J. A., & Rodríguez-Covili, J. (2011). Coordinating loosely-coupled work in construction inspection activities. *Group Decision and Negotiation*, 20(1), 39–56.
- Oliver, D. R. P., Demiris, G., Day, M., Courtney, K. L., & Porock, D. (2006). Telehospice support for elder caregivers of hospice patients: Two case studies. *Journal of Palliative Medicine*, 9(2), 264–267.
- OMG. (2014). *Business Process Model and Notation Specification*. <https://www.omg.org/spec/BPMN/>
- OMG. (2015). *UML Specification*. <https://www.omg.org/spec/UML/2.5/About-UML/>
- OMG. (2016). *Case Management Model and Notation Specification*. <https://www.omg.org/spec/CMMN>
- Osguthorpe, G., Steele, B., & Houldcroft, A. (1996). A method for developing CSCW systems. *Proceedings of WET ICE '96. IEEE 5th Workshop on Enabling Technologies; Infrastructure for Collaborative Enterprises*, 214–219. <https://doi.org/10.1109/ENABL.1996.555224>
- Pekkola, S., Kaarilahti, N., & Pohjola, P. (2006). Towards formalised end-user participation in information systems development process: Bridging the gap between participatory design and ISD methodologies. *Proceedings of the Ninth Conference on Participatory Design: Expanding Boundaries in Design-Volume 1*, 21–30.
- Pinelle, D. (2004). *Improving Groupware Design for Loosely Coupled Groups* [PhD Thesis]. Citeseer.
- Pinelle, D., & Gutwin, C. (2005). A groupware design framework for loosely coupled workgroups. *ECSCW 2005: Proceedings of the Ninth European Conference on Computer-Supported Cooperative Work*, 65–82.

- Pinelle, D., & Gutwin, C. (2006). Loose Coupling and Healthcare Organizations: Deployment Strategies for Groupware. *Computer Supported Cooperative Work (CSCW)*, 15(5), 537–572. <https://doi.org/10.1007/s10606-006-9031-2>
- Pinquart, M., & Sörensen, S. (2011). Spouses, adult children, and children-in-law as caregivers of older adults: A meta-analytic comparison. *Psychology and Aging*, 26(1), 1–14. <https://doi.org/10.1037/a0021863>
- Plöthner, M., Schmidt, K., de Jong, L., Zeidler, J., & Damm, K. (2019). Needs and preferences of informal caregivers regarding outpatient care for the elderly: A systematic literature review. *BMC Geriatrics*, 19(1), 1–22. <https://doi.org/10.1186/s12877-019-1068-4>
- Portillo-Rodríguez, J., Vizcaíno, A., Piattini, M., & Beecham, S. (2012). Tools used in Global Software Engineering: A systematic mapping review. *Information and Software Technology*, 54(7), 663–685. <https://doi.org/10.1016/j.infsof.2012.02.006>
- Robinson, H., Segal, J., & Sharp, H. (2007). Ethnographically-Informed Empirical Studies of Software Practice. *Inf. Softw. Technol.*, 49(6), 540–551. <https://doi.org/10.1016/j.infsof.2007.02.007>
- Rodríguez-Covili, J., Ochoa, S. F., Pino, J. A., Herskovic, V., Favela, J., Mejía, D., & Morán, A. L. (2011). Towards a reference architecture for the design of mobile shared workspaces. *Future Generation Computer Systems*, 27(1), 109–118.
- Routis, I., Nikolaidou, M., & Anagnostopoulos, D. (2018). Modeling collaborative processes with CMMN: success or failure? An experience report. In *Enterprise, Business-Process and Information Systems Modeling* (pp. 199–210). Springer.
- Runeson, P., & Höst, M. (2008). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2), 131. <https://doi.org/10.1007/s10664-008-9102-8>

- Savolainen, P., Ahonen, J. J., & Richardson, I. (2015). When Did Your Project Start? - The Software Supplier's Perspective. *J. Syst. Softw.*, 104(C), 32–40. <https://doi.org/10.1016/j.jss.2015.02.041>
- Schmid, K. (2002). A Comprehensive Product Line Scoping Approach and Its Validation. *Proceedings of the 24th International Conference on Software Engineering*, 593–603. <https://doi.org/10.1145/581339.581415>
- Schuler, D., & Namioka, A. (1993). *Participatory design: Principles and practices*. CRC Press.
- Seybold, C., Glinz, M., & Meier, S. (2005). Simulation-based validation and defect localization for evolving, semi-formal requirements models. *12th Asia-Pacific Software Engineering Conference (APSEC'05)*, 10 pp.-. <https://doi.org/10.1109/APSEC.2005.97>
- Shanks, G., & Darke, P. (1999). Understanding corporate data models. *Information & Management*, 35(1), 19–30.
- Shmueli, O., & Ronen, B. (2017). Excessive software development: Practices and penalties. *International Journal of Project Management*, 35(1), 13–27. <https://doi.org/10.1016/j.ijproman.2016.10.002>
- Smith, R. B., Hixon, R., & Horan, B. (1998). Supporting Flexible Roles in a Shared Space. *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, 197–206. <https://doi.org/10.1145/289444.289494>
- Sommerville, I. (2015). *Software Engineering*. Pearson Education. https://books.google.com.mx/books?id=x_qZCgAAQBAJ
- Spanoudakis, G., & Zisman, A. (2001). Inconsistency management in software engineering: Survey and open research issues. In *Handbook of Software Engineering and Knowledge Engineering: Volume I: Fundamentals* (pp.

- 329–380). World Scientific.
- Spitzberg, B. H. (2006). Preliminary development of a model and measure of computer-mediated communication (CMC) competence. *Journal of Computer-Mediated Communication*, 11(2), 629–666.
- Sutcliffe, A. (2010). Collaborative Requirements Engineering: Bridging the Gulfs Between Worlds. In S. Nurcan, C. Salinesi, C. Souveyet, & J. Ralyté (Eds.), *Intentional Perspectives on Information Systems Engineering* (pp. 355–376). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-12544-7_20
- Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. *Learning and Instruction*, 4(4), 295–312. [https://doi.org/10.1016/0959-4752\(94\)90003-5](https://doi.org/10.1016/0959-4752(94)90003-5)
- Szyperski, C., Gruntz, D., & Murer, S. (2002). *Component software: Beyond object-oriented programming*. Pearson Education.
- Tack, G.-R., Choi, M.-H., Lee, S.-J., Yang, J.-W., Kim, J.-H., Choi, J.-S., Jun, J.-H., Lee, J.-W., Park, J.-Y., Moon, S.-W., & Chung, S.-C. (2011). Prioritizing problem features in Korean patients with senile dementia for implementation of monitoring technologies. *Psychiatry Research*, 187(3), 418–423. <https://doi.org/10.1016/j.psychres.2010.06.031>
- Taherdoost, H., & Abolfazl, K. (2018). Managing Successful IT Project; Marketing Perspective. *Recent Advances on Computational Science and Applications*, 28(11), 144–153.
- Tan, M. (1994). Establishing Mutual Understanding in Systems Design: An Empirical Study. *J. Manage. Inf. Syst.*, 10(4), 159–182. <https://doi.org/10.1080/07421222.1994.11518024>
- Tang, J. C. (1991). Findings from observational studies of collaborative work.

International Journal of Man-Machine Studies, 34(2), 143–160.

Taylor, R. N., Medvidovic, N., & Dashofy, E. (2009). *Software Architecture: Foundations, Theory, and Practice*. John Wiley & Sons.

The Open Group. (2019). *Archimate Specification*.
<https://pubs.opengroup.org/architecture/archimate3-doc/>

Turkman, S., & Taweel, A. (2019). Business Process Model Driven Automatic Software Requirements Generation. In B. Shishkov (Ed.), *Business Modeling and Software Design* (pp. 270–278). Springer International Publishing.

Uber Company. (2022). *How Uber Works for Drivers and Riders*. Uber Technologies Inc.
<https://www.uber.com/us/en/about/how-does-uber-work> [Accessed 28 December 2022]

Uhl, N. P. (1983). *Using Research for Strategic Planning*. Jossey-Bass.

Vera, T., Ochoa, S. F., & Perovich, D. (2021a). An Instrument to define the Product Scope at Pre-selling Time. *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 604–608.
<https://doi.org/10.1109/CSCWD49262.2021.9437717>

Vera, T., Ochoa, S. F., & Perovich, D. (2021b). Requirements Engineering in the Pre-Contract Stage: Exploring the Processes and Practices Used in Small and Medium-Sized Software Enterprises. *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, 1346–1353.
<https://doi.org/10.1145/3412841.3442009>

Vessey, I. (1991). Cognitive Fit: A Theory-Based Analysis of the Graphs Versus Tables Literature*. *Decision Sciences*, 22(2), 219–240.
<https://doi.org/10.1111/j.1540-5915.1991.tb00344.x>

Weber, R., Coopers & Lybrand, & Accounting Association of Australia and New Zealand.

- (1997). *Ontological foundations of information systems*. Coopers & Lybrand and the Accounting Association of Australia and New Zealand.
- Weißberger, B., & Vogel-Heuser, B. (2012). Evaluation of a graphical modeling language for the specification of manufacturing execution systems. *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, 1–7. <https://doi.org/10.1109/ETFA.2012.6489669>
- Wood, J., & Silver, D. (1995). *Joint Application Development (2nd Ed.)*. John Wiley & Sons, Inc.
- Woolridge, R. W., Hale, D. P., Hale, J. E., & Sharpe, R. S. (2009). Software Project Scope Alignment: An Outcome-Based Approach. *Commun. ACM*, 52(7), 147–152. <https://doi.org/10.1145/1538788.1538822>
- Wu, M., Bhowmick, A., & Goldberg, J. (2012). Adding structured data in unstructured web chat conversation. *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, 75–82.
- Yin, R. K. (2008). *Case Study Research: Design and Methods (Applied Social Research Methods)* (Fourth Edition.). Sage Publications.
- Zhu, H., & Zhou, M. (2006). Role-based collaboration and its kernel mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(4), 578–589. <https://doi.org/10.1109/TSMCC.2006.875726>
- Zowghi, D., & Coulin, C. (2005). Requirements elicitation: A survey of techniques, approaches, and tools. In *Engineering and managing software requirements* (pp. 19–46). Springer.

Annexes

Annex A. Instrument for Stakeholder - Replica I

INSTRUCTIONS: Please, answer the following sentences according to your perception in the last meeting with the engineers

1. Do you consider that your brother (the person supporting the care of your mom) is represented in some role within the diagram? Please identify such a role.
2. Do you consider that your sister (the person in touch with the doctor) is represented in some role within the diagram? Please identify such a role.
3. Do you consider that the employee (the person assisting your mom) is represented in some role within the diagram? Please identify such a role.
4. Which communication services (e.g., text messaging) are available to you in the application to interact with your siblings?
5. What services does the application provide to the employee to communicate with you?
6. Will the proposed application allow you to communicate with the doctor who is in touch with your sister? Please justify your answer.
7. Do you consider that the diagram represents the communication needs of the participants? If not, please indicate what things you would change.
8. Do you consider that using this kind of diagram helps explain the communication needs of the participants?
9. Do you consider that the prototype helps you understand how an application could support you in this process? If not, please indicate what things you would change or add to the prototype.
10. Do you consider that the engineers interviewing you understood the communication needs of the participants?

Annex B. Instrument for Engineers - Replica I

INSTRUCTIONS: Please, answer the following sentences according to your perception in the last meeting with the stakeholder

1. Do you consider that the stakeholder's brother is represented in some role in the diagram? Please identify such a role.
2. Do you consider that the stakeholder's sister (the person in touch with the doctor) is represented in some role in the diagram? Please identify such a role.
3. Do you consider that the employee (the person assisting the elderly) is represented in some role in the diagram? Please identify such a role.
4. Which communication services (e.g., text messaging) are available to the stakeholder in the application to interact with her siblings?
5. What services does the application provide to the employee to communicate with the stakeholder?
6. Will the proposed application allow the stakeholder to communicate with the doctor who is in touch with the sister? Please justify your answer.
7. Do you consider that the diagram represents the communication needs of the participants? If not, please indicate what things you would change.
8. Do you consider that using this kind of diagram helps explain the communication needs of the participants?
9. Do you consider that the prototype helps the stakeholders understand how an application could support participants in this process? If not, please indicate what things you would change or add to the prototype.
10. Do you consider that using the stakeholder dialect really contributes to achieving a shared understanding about the scenario with the stakeholder?
11. After this process, how clear were the stakeholder's requirements to you? [Not clear at all, not very clear, more or less clear, clear, completely clear]

Annex C. Instrument for Stakeholder - Replica II

INSTRUCTIONS: Please, answer the following sentences according to your perception in the last meeting with the engineers

1. Do you consider that your husband is represented in some role within the diagram? Please identify such a role.
2. Do you consider that your children are represented in some role within the diagram? Please identify such a role.
3. Do you consider that the doctor is represented in some role within the diagram? Please identify such a role.
4. Which communication services (e.g., text messaging) are available to you in the application to interact with your husband?
5. What services does the application provide to your children to communicate with you?
6. Will the proposed application allow you to communicate with the doctor? Please justify your answer.
7. Do you consider that the diagram represents the communication needs of the participants? If not, please indicate what things you would change.
8. Do you consider that using this kind of diagram helps explain the communication needs of the participants?
9. Do you consider that the prototype helps you understand how an application could support you in this process? If not, please indicate what things you would change or add to the prototype.
10. Do you consider that the engineers interviewing you understood the communication needs of the participants?

Annex D. Instrument for Engineers - Replica II

INSTRUCTIONS: Please, answer the following sentences according to your perception in the last meeting with the stakeholder.

1. Do you consider that the stakeholder's husband is represented in some role in the diagram? Please identify such a role.
2. Do you consider that the stakeholder's children are represented in some role in the diagram? Please identify such a role.
3. Do you consider that the doctor is represented in some role in the diagram? Please identify such a role.
4. Which communication services (e.g., text messaging) are available to the stakeholder in the application to interact with her husband?
5. What services does the application provide to the siblings to communicate with the stakeholder?
6. Will the proposed application allow the stakeholder to communicate with the doctor? Please justify your answer.
7. Do you consider that the diagram represents the communication needs of the participants? If not, please indicate what things you would change.
8. Do you consider that using this kind of diagram helps explain the communication needs of the participants?
9. Do you consider that the prototype helps the stakeholders understand how an application could support participants in this process? If not, please indicate what things you would change or add to the prototype.
10. Do you consider that using the stakeholder dialect really contributes to achieving a shared understanding about the scenario with the stakeholder?
11. After this process, how clear were the stakeholder's requirements to you? [Not clear at all, not very clear, more or less clear, clear, completely clear]

Annex E. Instrument for Stakeholder - Replica III

INSTRUCTIONS: Please, answer the following sentences according to your perception in the last meeting with the engineers

1. Do you consider that your sister is represented in some role within the diagram? Please identify such a role.
2. Do you consider that your children are represented in some role within the diagram? Please identify such a role.
3. Do you consider that the doctor is represented in some role within the diagram? Please identify such a role.
4. Which communication services (e.g., text messaging) are available to you in the application to interact with your sister?
5. What services does the application provide to your sister to communicate with you?
6. Will the proposed application allow you to communicate with the doctor? Please justify your answer.
7. Do you consider that the diagram represents the communication needs of the participants? If not, please indicate what things you would change.
8. Do you consider that using this kind of diagram helps explain the communication needs of the participants?
9. Do you consider that the prototype helps you understand how an application could support you in this process? If not, please indicate what things you would change or add to the prototype.
10. Do you consider that the engineers interviewing you understood the communication needs of the participants?

Annex F. Instrument for Engineers - Replica III

INSTRUCTIONS: Please, answer the following sentences according to your perception in the last meeting with the stakeholder

1. Do you consider that the stakeholder's sister is represented in some role in the diagram? Please identify such a role.
2. Do you consider that the stakeholder's children are represented in some role in the diagram? Please identify such a role.
3. Do you consider that the medic is represented in some role in the diagram? Please identify such a role.
4. Which communication services (e.g., text messaging) are available to the stakeholder in the application to interact with her sister?
5. What services does the application provide to the sister to communicate with the stakeholder?
6. Will the proposed application allow the stakeholder to communicate with the doctor? Please justify your answer.
7. Do you consider that the diagram represents the communication needs of the participants? If not, please indicate what things you would change.
8. Do you consider that using this kind of diagram helps explain the communication needs of the participants?
9. Do you consider that the prototype helps the stakeholders understand how an application could support participants in this process? If not, please indicate what things you would change or add to the prototype.
10. Do you consider that using the stakeholder dialect really contributes to achieving a shared understanding about the scenario with the stakeholder?
11. After this process, how clear were the stakeholder's requirements to you? [Not clear at all, not very clear, more or less clear, clear, completely clear]

Annex G. Instrument for Stakeholder - Replica IV

INSTRUCTIONS: Please, answer the following sentences according to your perception in the last meeting with the engineers

1. Do you consider that your mother-in-law is represented in some role within the diagram? Please identify such a role.
2. Do you consider that your husband is represented in some role within the diagram? Please identify such a role.
3. Do you consider that the doctor is represented in some role within the diagram? Please identify such a role.
4. Which communication services (e.g., text messaging) are available to you in the application to interact with your mother-in-law?
5. What services provide the application to your mother-in-law for communicating with you?
6. Will the proposed application allow you to communicate with the doctor? Please justify your answer.
7. Do you consider that the diagram represents the communication needs of the participants? If not, please indicate what things you would change.
8. Do you consider that using this kind of diagram helps explain the communication needs of the participants?
9. Do you consider that the prototype helps you understand how an application could support you in this process? If not, please indicate what things you would change or add to the prototype.
10. Do you consider that the engineers interviewing you understood the communication needs of the participants?

Annex H. Instrument for Engineers - Replica IV

INSTRUCTIONS: Please, answer the following sentences according to your perception in the last meeting with the stakeholder

1. Do you consider that the stakeholder's mother-in-law is represented in some role in the diagram? Please identify such a role.
2. Do you consider that the stakeholder's husband is represented in some role in the diagram? Please identify such a role.
3. Do you consider that the medic is represented in some role in the diagram? Please identify such a role.
4. Which communication services (e.g., text messaging) are available to the stakeholder in the application to interact with her mother-in-law?
5. What services does the application provide to the mother-in-law to communicate with the stakeholder?
6. Will the proposed application allow the stakeholder to communicate with the doctor? Please justify your answer.
7. Do you consider that the diagram represents the communication needs of the participants? If not, please indicate what things you would change.
8. Do you consider that using this kind of diagram helps explain the communication needs of the participants?
9. Do you consider that the prototype helps the stakeholders understand how an application could support participants in this process? If not, please indicate what things you would change or add to the prototype.
10. Do you consider that using the stakeholder dialect really contributes to achieving a shared understanding about the scenario with the stakeholder?
11. After this process, how clear were the stakeholder's requirements to you? [Not clear at all, not very clear, more or less clear, clear, completely clear]

Annex I. Analysis of Notations for Modeling User Interaction Scenarios in Ubiquitous Collaborative Systems

Next, an extract of the paper named “Analysis of Notations for Modeling User Interaction Scenarios in Ubiquitous Collaborative Systems” (Canché et al., 2019) is described. Here, a comparative study on the usability and usefulness of MCM and two versions of CIMoN is presented to help software designers determine the probability of adoption of these modeling languages and also identify the aspects susceptible to be improved.

I.1. Computer-mediated User Interaction Modeling Notations

The three notations to be analyzed (i.e., MCM and CIMoN 1 and 2) are based on a graph, where the nodes represent the roles played by the actors participating in the collaborative process, and the edges indicate the potential interactions (links) among them. These links are always represented between two roles to reduce the complexity of specifying the interaction scenarios; however, combinations of these links can be specified at a higher level (e.g., at coordination level) in order to support simultaneous relations between more than two roles.

The graphs resulting from modeling the interaction scenarios are named *Roles Interaction (RI) graphs* (Herskovic et al., 2009). In order to ease the explanation of these notations and also the interaction scenarios that can be represented with them, next we briefly describe a simulated scenario that will be used to illustrate the expressiveness of the notations under study. Then, we describe the representation of roles and links in the three notations, and analyze their expressiveness, usefulness and usability.

I.1.1. Simulated Interaction Scenario

The simulated scenario is inspired by regular urban emergency response processes conducted by firefighters, for instance, during a fire in a building. As mentioned before, the actors are conceived as roles (i.e., user profiles) and represented as nodes in the RI graph. Figure I.1 shows a simplified version of a RI graph representing such an interaction scenario using box-and-lines.

According to the graph, in this people-driven process participate first responders playing four possible roles: *incident commander* (in charge of the whole operation in the field), *search and rescue officer* (in charge of a team that rescues people), *paramedic* (provides first-aids and transportation to injured people) and *logistic officer* (provides tools to other first responders to perform their activities). The RI graph shows that the incident commander needs to interact with every role available in the field to coordinate their response activities. Moreover, the search and rescue officers interact directly with the paramedic due to their activities are tightly linked; consequently, the collaboration between them has time constraints. For the same reason, search and rescue officers

leading different teams need to interact directly among them. In this case, the roles that are present in urban emergency response processes, their responsibilities and the interaction protocols among them come from the fire fighting norms and search and rescue protocols. This information is used as input to specify a generic interaction scenario that should be supported during the people-driven processes. Any software system designed to support activities in such a process must consider that interaction scenario.

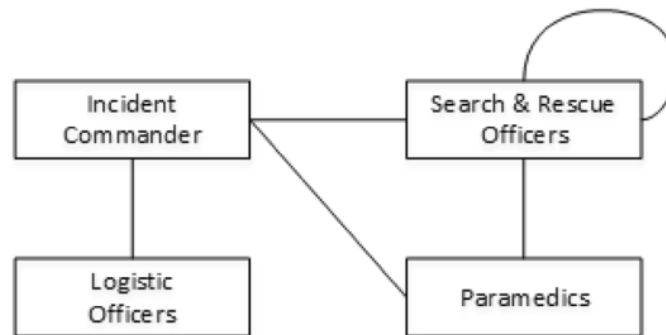


Figure I.1. Interaction scenario in an urban emergency response process (simplified version).

Typically, the specification of the interaction scenario only shows the participating roles (nodes) and their needs of supporting interactions with other roles (link between nodes). Although this information is useful to understand the interaction requirements to be addressed by a collaborative system that assists the people in such a process (or in part of it), additional information is required to decide which services would be more appropriate to support these interactions. For instance, knowing the availability of the target user (i.e., the availability of his/her role), at the time that other user needs to collaborate with the former, will help determine if the interaction service should be synchronous, asynchronous or if both of them should be provided by the supporting system.



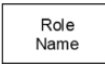
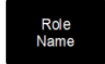

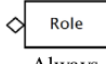
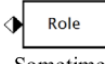
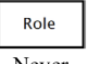

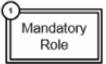
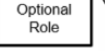
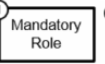
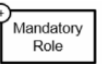
Due to the need of making this type of decisions at design time, the notations to be used for representing the RI graphs should allow representing additional information about the roles and the links between them. Next, we introduce the analyzed notations and indicate how they provide additional information that helps designers make these decisions.

I.1.2. Modeling Actors

As mentioned before, in the three notations the actors are conceived as roles (i.e., user profiles) and represented as nodes in the RI graph. The node's name determines the role played by such an actor (i.e., his/her responsibilities), and the node's shape indicates the type of participant that will play such a role. The three notations consider

the participation of *human* and *intermediary actors* (Table I.1). The former are people that use the system services to play their role (e.g., the incident commander), and it is assumed that they are able to understand their work contexts and take actions accordingly. The intermediary actors are passive software components that provide data persistence and answer data requests from other actors (e.g., a shared data repository deployed at the command post to ease the information exchange among first responders). Additionally, CIMoN allows specifying *autonomous agents* as actors able to play roles in people-driven processes, which provide more expressiveness to the notation. These agents are typically used to deliver automatic notifications (e.g., warning or alarms) to the participants in these processes.

Table I.1. Comparison of expressiveness aspects of MCM and CIMoN.

Concept	MCM	CIMoN 1	CIMoN 2
Actor type	 Role Name Human Actor  Role Name Intermediary	 Role Name Human Actor  Role Name Intermediary  Role Name Autonomous Agent	
Actor availability	---	 Role Always Available  Role Sometimes Available  Role Never Available	
Minimum arity (mandatoriness) of an actor	---	 Optional Role  Mandatory Role	 Optional Role  Mandatory Role  Mandatory Role

Some types of actors allow us to expect a certain availability when other roles try to interact with them; specifically the actors can be *always*, *sometimes* and *never available*. For instance, we can expect that intermediary actors and autonomous agents are available to interact with others when required. However, this cannot be assumed for human actors, since they manage their own workflow and their availability to interact with others. For instance, the incident commander could be busy when a logistics officer intends to interact with him, making such an attempt unsuccessful.

The roles' availability is represented through a visual attribute of the node. Only CIMoN allows specifying the availability of actors, which is specified with different rhombus attached to the actor as shown in Table I.1. The availability does not characterize the actor itself but its participation in an interaction (i.e., in a link), as we can expect different availability depending on the interaction priorities given by a role to certain interaction requests. For instance, the incident commander could be always available for the leaders of search and rescue teams, but sometimes available for logistic officers.

CIMoN 1 and 2 also allows representing the minimum arity or cardinality of the roles required to conduct a people-driven collaborative process; i.e., it is possible to specify if a certain role is mandatory, and if it does, it indicates how many instances of such a role (actors) are required at least. This attribute is specified with a circle in the upper left

corner of the role, and its representation was refined in CIMoN 2. Optional roles are shown with an empty circle or without this symbol. Other possible values for this attribute are “1” (indicating that at least one instance of such a role is required) and “+” (indicating that at least two instances are required). For instance, in the urban emergency response process the minimum arity for the incident commander role is 1, since such a role is mandatory in every process; however, counting on logistic officers or paramedics is optional.



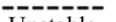

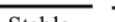
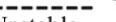

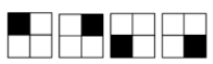

Summarizing, in terms of roles representations, CIMoN 2 is a bit more expressive than CIMoN 1 and much more than MCM. In the next section we repeat this analysis, but for the links between actors.

I.1.3. Modeling links between actors

All interaction requirements in the three notations are specified with a line between the involved roles. In case of MCM, it only represents the existence of an interaction need between two roles (Table I.2), while in CIMoN 1 and 2 the designer can use that line to characterize the stability of the communication link. Such a feature of the link represents the feasibility to count on a stable communication channel between the participants at the time they decide to collaborate. The link stability can be characterized as *stable* (solid line), *unstable* (dashed line) or *uncommunicated* (a partial line between these actors in CIMoN 1, and a dashed line with two crossing bars in CIMoN 2). This concept is close to the *reachability* aspect introduced in MCM (Herskovic et al. 2009). The technology used to support these interactions determines in part the stability of these links and feasibility to establish an effective communication when the participants need to interact among them and are available to do it. This is one of the reasons why firefighters typically use radio systems to exchange information through voice messages.

Regardless of the specification of the links stability, the edges include one or more labels that represent the type of interaction that is required between the actors; i.e., these are interaction requirements that must be addressed by the supporting systems. The type of interactions can be synchronous, asynchronous or both. In MCM, it is represented through a matrix with *colored quadrants* that combines the type of synchronicity support that is required, and the reachability (similar to the stability of the communication link) between the involved roles. In both versions of CIMoN this feature is specified with a label on each link. The label can indicate that an “S” (*synchronous*), “A” (*asynchronous*) or “SA” (*synchronous and asynchronous*) support is required between the two involved roles. In this sense, the representations of the link type and roles interaction requirements are mixed in MCM and differentiated in CIMoN. In case of the urban emergency processes, the activities performed by the roles and the needs of interaction among them determine the type of synchronicity that is required between every couple of roles.

Table I.2. Representation of links between actors.

Concept	MCM	CIMoN 1			CIMoN 2		
Types of links between actors	 Interaction Required	 Stable Link	 Unstable Link	 No Link	 Stable Link	 Unstable Link	 No Link
Requirement of interaction support		S (synchronous), A (asynchronous), SA (synchronous and asynchronous)					
Actor availability	---						
Cardinality of the Interaction	---	---			* (all-to-all), + (various-to-various), 1 (one-to-one)		

As indicated in the previous section, CIMoN 1 and 2 allow representing (through different rhombus) the actors' availability. This attribute is a property of both, roles and links, since it is initially set during the roles definition stage, and then it can be updated when the designer specifies the links between roles (for instance, due to the introduction of priority in the interactions among actors).

Although the three notations allow specifying interactions between two roles, the interaction process can involve various actors per role. In this sense, CIMoN 2 includes the concept of cardinality of the interaction that is represented with a symbol on the link. These symbols can be "*" that indicates at most *all-to-all* interactions (which also includes *various-to-all*, *one-to-all*, and vice versa), "+" that specifies at most *various-to-various* interactions (which also includes *one-to-various* and vice versa) and "1" or no symbol that indicates exactly *one-to-one* collaborations. In case of emergency responses, the incident commander has to count on the capability to deliver messages in the three ways since it is required to keep the activities coordination in the field. However, other roles like the paramedics or logistic officers do not require (and usually it is not allowed) to count on capability to send broadcast messages to first responders (i.e., to perform many-to-many interactions), since the control and responsibility of the process is centralized in the incident commander.

Once again, we can see that the notation proposed by CIMoN 2 is richer than the other two options; however, this will be useful for system designers only if they perceive the extra expressiveness as a valuable asset, and they are able to use it in practice. The next section describes the study conducted with system designers that tries to understand the trade-off between usability and expressiveness of these notations.

I.2. Analyzing the Trade-off between Usability and Usefulness of the Studied Notations

The cognitive effectiveness of a visual notation is defined as the speed, ease and accuracy with which a representation can be processed by the human mind (Larkin and Herbert, 1987), and this aspect affects the understandability and usefulness of the notations. In order to characterize the effect on cognitive effectiveness when augmenting the expressiveness of a notation, we conducted a pilot study that used MCM as baseline for understanding such an effect. The study involved two work sessions with software designers, who empirically determined the usability and usefulness of each version of CIMoN, comparing them with MCM. Moreover, we conducted a follow up interview with a sample of the participants in order to understand more in depth the experimental results.

Next we describe the general settings of the evaluation activities, and present and discuss the obtained results. Then, the results of a follow-up interview are presented to help understand the trade-off between usability and expressiveness (usefulness) of the analyzed notations. Finally, we present the threats to validity.

I.2.1. General Settings

The participants in this evaluation were computer science graduate students, most of them enrolled in the doctorate program delivered by the Computer Science Department of the University of Chile. The evaluation involved two work sessions; sixteen people participated in the first one (WS1) and thirteen in the second one (WS2). The WS1 considered four women and twelve men, in the range of 24-36 years old. The WS2 included five women and eight men, in the range of 28-36 years old.

All participants were knowledgeable in visual modeling and processes representations, but not necessarily in specifying collaborative processes or developing collaborative systems. Moreover, these people never used the MCM or CIMoN notations before this exercise.

The participants were recruited using a snowballing strategy, their participation was voluntary and none of them had work or study relationships with the authors. These participants were opportunely informed about the possibility to quit the evaluation process whenever they wanted, and without consequences.

In these sessions, the participants should specify the interaction scenarios that are present in the process for elaborating their thesis proposal and one of the authors acted as moderator. Such a process, which was well-known by the participants, considers several activities, roles and interactions among roles. As there is not a pre-established workflow to be followed by the actors, the process can be considered people-driven and

collaborative.

The evaluators participating in the first work session used MCM and CIMoN 1 to specify the interaction scenarios, and those who participated in the second one used MCM and CIMoN 2. In order to reduce the learning bias, half of the participants started the modeling process using MCM, and the other half started with CIMoN.

At the beginning of each session, the participants were asked to indicate general information about them into the form received to specify the interaction scenarios. Such information included gender, age, level of experience modeling processes (in general, and also collaborative), and also experience developing collaborative systems. After that, the moderator instructed the participants about the foundations and representations of interaction models using MCM and CIMoN. Then, the moderator presented a simplified version of the collaborative process to be modeled, using slides with the visual representation for storytelling proposed in (Antunes et al., 2013).

The next step was the modeling process conducted by the participants, who started using the notation assigned to them, and then followed by specifying the same interaction scenario but using the other notation. Both models were specified in the form the participants received at the beginning of the session. All forms included a numeric ID that was unique for each participant, who also indicated the beginning and ending time of their modeling activity using each notation.

After this modeling activity, the participants filled out a questionnaire that allowed us to gather information about the usability and usefulness of both notations. Moreover, they provided qualitative information, through comments, at the end of the questionnaire.

1.2.2. Analyzing the Correctness and Completeness of the Models

The participants specified between eight and ten different interaction scenarios in their models (i.e., RI graphs). These specifications were then compared to a rubric to determine their completeness and correctness, and based on it, a score was assigned to each model. These scores ranged from 1 to 5: (1) very poor, (2) poor, (3) acceptable, (4) good, and (5) very good.

The results of this modeling process show that all participants were able to create at least acceptable RI graphs, in terms of their correctness and completeness, using both notations (Table I.3). No models were rated with unacceptable scores, and the average score was between 3.7 and 3.9 for both notations in both work sessions. While not significantly different, it is noticeable that even though CIMoN is more expressive, making more constructs available to the designers, models specified with such a notation tend to be more correct than those specified with MCM. The results obtained on this last aspect were significantly different ($\alpha = 0.05$) for the WS1, but no for the WS2.

Table I.3. Scores assigned to the interaction models created by the participants.

Notation/Correctness		1. Very poor (%)	2. Poor (%)	3. Acceptable (%)	4. Good (%)	5. Very good (%)	Avg. correctness
WS1	MCM	0	0	44	31	25	3.8
	CIMoN 1	0	0	38	25	37	3.9
WS2	MCM	0	0	46	39	15	3.8
	CIMoN 2	0	0	31	46	23	3.9

The average time spent to carry out the modeling activity using MCM were 8 and 8.9 minutes in WS1 and WS2 respectively. This time was 9.3 minutes when using CIMoN 1, and it grew up to 13.4 minutes with CIMoN 2. When analyzing the mean values between both models in each evaluation scenario, we identified a statistically significant difference in WS2: $t(12) = 3.0955$, $p < 0.01$, $d = 0.86$. The difference between the mean values is not statistically significant in WS1.

These differences in the modeling time can be explained considering the number of symbols of each notation, that makes the participants learn and represent more information in the interaction scenarios. In particular, MCM has fewer symbols than CIMoN 1, and this latter has less symbols than CIMoN 2. This can explain the increasing time to represent the same model when designers used different notations.

The values obtained for models correctness and completeness, and also the times required to create these models indicate that the three notations are usable and useful enough; i.e., their cognitive effectiveness seems to be appropriate for designers of these systems. In this sense, it is important to take a step forward to understand how the additional expressiveness of a notation affects its learnability and usability, and consequently its usefulness. Next section analyzes such an aspect for the three notations.

I.2.3. Analyzing the Usability and Usefulness of the Notations

The approach used to evaluate the usability and usefulness of each notation was based on the goal-question-metric model (Basili et al., 1994). Such a model was used to conceive an evaluation questionnaire following the guidelines indicated in (Buchanan and Salako, 2009). The questionnaire was filled by all participants in the WS1 and WS2, immediately after specifying the models indicated in the previous section. Next, we present the questionnaire, the process followed to analyze the opinion of the participants, and the results obtained about the usability and usefulness of the visual notations.

I.2.3.1. Data Collection Instrument

The questionnaire used for the data collection considered five attributes, evaluating six metrics by means of a single item each, as shown in Table I.4. For each item, the participants specified a score ranging from 1 to 4: (1) strongly disagree, (2) disagree, (3)

agree, and (4) strongly agree. We used a four-point scale to make participants have to adopt a binary position (i.e., positive or negative) when rating each attribute of a notation. This rating strategy was also suggested by Buchanan and Salako (2009) to evaluate usability and usefulness of visual artifacts.

Table I.4. Instrument used to evaluate the usability and usefulness of a visual notation.

Attribute	Item (asks if...)	Metric
Effectiveness	1. The interaction model carried out is complete	Expressiveness
Aesthetic appearance	2. Nice visual appearance	Attractiveness
	3. The notation symbols are simple for using	Simplicity
Understandability	4. The used symbols and labels are understandable	Comprehension
Intuitiveness	5. The used symbols are intuitive	Intuitiveness
Learnability	6. It was easy and quick learn to use the notation	Easiness to learn

I.2.3.2. Data Analysis and Results

In order to study whether there are statistically significant differences between the six evaluation metrics described in Table I.4, we ran Wilcoxon signed-rank tests, comparing the median values of the reported scores by study participants in scenarios WS1 and WS2. The choice of a non-parametric statistic obeys to the fact that data failed to satisfy the normality assumption (verified through a Shapiro-Wilk test), and it being expressed in ordinal scale. Statistical analyses were run using R version 3.5.1. Table I.5 summarizes the statistical results. A star (*) indicates a significant difference for $\alpha = 0.05$ and a double star (**) indicates a significant difference for $\alpha = 0.01$.

Considering the statistical significance of the results we can see two different, but complementary findings. On the one hand, in the WS1 the participants found that CIMoN is more expressive and intuitive than MCM, which makes sense since CIMoN was designed particularly with such a purpose. On the other hand, in the WS2 the participants indicated that MCM is more simple and easy to learn than CIMoN, which can also be explained by the fact that MCM has less symbols than CIMoN that reduce the cognitive load required to use it.

The findings were also supported by comments provided by the participants in the questionnaire form. For instance, four participants stated that MCM is simpler and easier to understand with regard to CIMoN, but in MCM it was not possible to specify several aspects of the process. In addition, five participants declared that CIMoN is more expressive than MCM. Moreover, two participants mentioned that MCM is easier to learn than CIMoN, but they found that none of these notations is intuitive. As expected, the results show that expressiveness and learnability of a notation tends to be antagonistic properties; therefore, increasing the expressiveness of the notation will negatively affect its learnability.

Table I.5. Differences in median values for evaluation metrics (Wilcoxon signed-rank

Scenario	Metric	Median MCM	Median CIMoN	W	p	Z	r
WS1	Expressiveness	3	3.5	0	*	-2.441	0.61
	Attractiveness	3.5	3.5	12	-	0.090	0.02
	Simplicity	3	3	14.5	-	-0.644	0.16
	Comprehension	3	3	2	-	-1.400	0.35
	Intuitiveness	2.5	3	0	*	-2.626	0.66
	Easiness to learn	4	3.5	22	-	1.241	0.31
WS2	Expressiveness	3	3	3	-	-1.663	0.46
	Attractiveness	4	3	42	-	1.688	0.47
	Simplicity	4	3	81	*	2.581	0.72
	Comprehension	3	3	21	-	1.201	0.33
	Intuitiveness	3	3	37	-	1.777	0.49
	Easiness to learn	4	3	66	**	3.138	0.87

tests).

Although no contradictions were identified in the results of each work session and both findings were aligned, we intended to understand the difference between the emphasis done by the participants in both work sessions. Clearly, the opinion of the participants in the WS1 highlighted the relevance of counting on expressive (useful) notations, while the participants in the WS2 put more emphasis on the usefulness of these notations.

After exploring and discarding several hypotheses to try to understand the difference between these two emphases, we identified an aspect that could explain it. Although all participants had similar backgrounds in modeling processes, and they specified models that were similar in terms of correctness and completeness, a third of the participants in WS1 had previous experience developing collaborative systems. These people recognized (in the questionnaire) the relevance of counting on symbols to specify the interaction scenario more in detail.

In order to understand more in depth these results we conducted follow-up individual interviews to a sample of the participants in every work session. Ten people were interviewed; five participants of each work session.

I.2.4. Follow-Up Interviews

The participants were randomly selected and the interview goals were two: 1) to determine the semantic transparency of the symbols considered in the three notations, which help understand the trade-off between usability and usefulness of these symbols, and 2) to identify reasons that explain the difference in the emphasis done by the participants to the usability and usefulness of the notations. Next we explain the process followed to explore these aspects and present the obtained results.

I.2.4.1. Analyzing the Semantic Transparency of the Symbols

In these interviews the participants evaluated the semantic transparency of the symbols

that they used in the previous modeling activity. The semantic transparency indicates how much the appearance of a symbol evokes the assigned meaning (Moody, 2009). In this sense, the values that a symbol can adopt are “intuitive”, “easy to learn”, “difficult to learn” or “confusing”. A symbol is intuitive if its visual representation evokes the assigned meaning. It is easy or difficult to learn depending on the cognitive effort required to associate the visual representation with its meaning. Finally, a symbol is confusing if its appearance evokes something different to the assigned meaning.

The interviews were individual. At the beginning, the participants received two forms, each with the symbols of the nomenclatures that they used during the previous modeling process. In such a form, they indicated the semantic transparency of each symbol using the previously explained scale. After rating all symbols for both nomenclatures, the interviewees were asked for reasons that make some symbols have low semantic transparency.

The results of the interviews indicate that there are no “intuitive” symbols in these notations; most of them are “easy to learn”, some of them are “difficult to learn”, and particularly one of them is “confusing” (the cardinality of the interaction in CIMoN 2). Table I.6 shows the category of symbols and the percentage of people that considered these symbols as “easy to learn”. The last two rows indicate the number of concepts and symbols of these notations. This provides a first insight of their expressiveness and capability of being used in practice.

Table I.6. Symbols rated as “easy to learn” in MCM and CIMoN.

Symbol	MCM	CIMoN 1	CIMoN 2
Actor type (roles)	100%	80%	80%
Types of links between actors	100%	100%	100%
Requirement of interaction support	30%	100%	100%
Actor availability	–	80%	80%
Minimum arity (mandatoriness) of an actor	–	70%	70%
Cardinality of the Interaction	–	–	0%
# of Concepts	3	5	6
# of Symbols	7	15	18

Concerning MCM, the results indicate the notation is perceived as affordable for designers, but the representation of interaction support (indicated with a matrix with colored quadrants as shown in Table I.2) is difficult to learn. This can explain why a more expressive notation (particularly CIMoN 1) is perceived at least as intuitive as MCM. The same effect was evidenced in CIMoN 2 due to the low semantic transparency of the symbols used to represent the cardinality of the interaction. In other words, the inclusion of a single symbol with low semantic transparency can jeopardize the usability and usefulness of the whole notation.

Concerning the two versions of CIMoN, the results confirm the existence of a trade-off between the expressiveness and usability of the notation. CIMoN 2 represents more concepts and includes more symbols than CIMoN 1, but it reduces its usability according to the designers' opinion.

I.2.4.2. Analyzing the Emphasis in the Usability or Usefulness of the Notation

As shown in Section I.2.3.2, the participants in the WS1 emphasized the value of counting on more expressive notations, while the participants in WS2 emphasized the usability of them. Considering these two perspectives, we tried to identify reasons for such a difference.

After asking for it to the participants in the follow-up sessions, we identified that they built the specification of the interaction scenarios with two different (but complementary) purposes. Most people assumed the specification as the instrument that allows creating a shared understanding with other designers and stakeholders about the interaction scenario to be supported. In this sense, the model specification allows to communicate, evaluate and refine the interaction scenario.

Other people went a step further, and assumed the specification as an instrument that not only helps reach the previous purpose, but also informs the design of the collaborative system to be developed; i.e., the specification would be used to identify the services that will be required to support the activities of the end-users. As expected, these people were mainly participants with previous experience in the development of collaborative systems; therefore, they were able to see the benefits of counting on a more expressive notation.

These two perspectives are valid and also recognized in the literature of visual notations design. While it might seem reasonable to conceive a notation to be understandable to stakeholders and use it also to communicate designs to developers (following the "lowest common denominator" principle), the cognitive load theory suggests that this may be incorrect (Sweller, 1994). Particularly, optimizing representations for non-experts can reduce their effectiveness for experts and vice versa; this is called the expertise reversal effect (Kalyuga et al., 2003). In this sense, Moody (2009) recommends defining different representations for specifying models that should be agreed with non-technical people (e.g., with stakeholders), from those that are going to be used only by technical experts. This suggests that MCM, but mainly CIMoN should separate the specification in two dialects; one for representing the information (interaction scenario) to be agreed with stakeholders, and the other to be used only by the development team. Thus, the first one should emphasize the usability of the notation, while the second one can take advantage of more complex but expressive representations.

I.2.5. Threats to Validity

Some aspects could represent potential limitations to this study. One potential threat to the validity of the correctness of the models is the way in which we perform the scoring. However, the possible bias was addressed by checking the results with a rubric agreed upon by the researchers.

We can also consider that the participation of one author as guide of the semi-structured interview could have biased the obtained results. However, the participation of such a person comes down to read each item of the questionnaire and clarify its meaning if required.

The fact that the participants were not formal designers of interactive systems might have affected the perceived usability and usefulness of these notations. However, all of them were knowledgeable in visual modeling and processes representations, with experience of work in the software industry.

We acknowledge that the small number of participants poses a threat to the generalization of the results. For that reason, as mentioned, we complemented the quantitative results using qualitative information that helped us understand the outcomes in a broader sense.

I.3. Discussion

Modeling interaction scenarios among actors of a people-driven process helps identify the interaction requirements to be considered in the development of the ubiquitous collaborative system that supports such a process. Few notations and languages have shown to be capable of representing these scenarios, but none of them has been broadly adopted by software designers. Moreover, there is not a comparison of the usability and usefulness of these notations that helps designers to choose the most suitable one.

In order to contribute address such a need, this work presents an analysis of three notations: MCM and two versions of CIMoN. The comparison made evident the trade-off between usability and expressiveness of these notations. Particularly, MCM was perceived as more usable, but CIMoN was more useful since it is more expressive. Moreover, the evaluation results indicated that people can use these specifications with (at least) two different purposes: 1) to generate a shared understanding between designers and stakeholders about the interaction scenario to be supported, or 2) to reach the previous goal but also inform the design of the supporting system. Therefore, the answer about what notation to choose depends on the purpose that follows the interaction scenario specification and also on other factors, like the skills of the designers or the capability of training them in the use of these notations.

More usable notations tends to be more suitable for novice designers due to the entry barrier is lower. Consequently, more expressive notations are recommended for

experienced designers or people that can be trained in the use of these specifications. Moreover, using more expressive notations is worthwhile if the resulting models are going to be processed by computing programs to produce valuable results; e.g., to obtain the interaction requirements to be considered in the development of an ubiquitous collaborative system. This result opens the possibility for visual notations designers to define flavors of these representations (e.g., notations with different abstraction levels) to be used by people with different design skills.

The study results also show the notations have to include almost exclusively “intuitive” and “easy to learn” symbols. A single symbol with low semantic transparency is enough to jeopardize the adoption of whole notation. This finding is valuable not only for software designers, but also for designers of modeling languages who have to carefully select the visual representations to ensure their semantic transparency.

The trade-off between usability and expressiveness is highly relevant during the adoption of a notation, since it determines the entry barrier. However, it is not an issue once the designers have learned the notation, and in that case, its expressiveness gets more relevance particularly because it allows creating more complex or large models. In this sense, the use of software modeling tools for managing and processing these representations, particularly those large or complex, can provide several advantages for designers.

The next step in this research initiative considers to create the algorithms that automatically derive the interaction requirements of an ubiquitous collaborative system, based on the analysis of these interaction models. Moreover, we plan to evaluate the expressiveness of the notations doing an exercise inverse to the one presented in this paper; specifically, to give interaction model representations to the designers (specified in the different notations), and ask them to infer the services to be provided by the supporting system being developed. This result will allow us to determine if the improvement of the notation expressiveness really generates a benefit for developers, since it helps them infer more services and design aspects relevant for the solution.

Annex J. List of Publications

The following papers were published as a result of this research:

- Canché, M., & Ochoa, S. F. (2018). Modeling Computer-Mediated User Interactions in Ubiquitous Collaborative Systems. In J. Bravo & O. Baños (Eds.), *Proceedings of the 12th International Conference on Ubiquitous Computing and Ambient Intelligence*, UCAml 2018, Punta Cana, Dominican Republic, December 4-7, 2018 (Vol. 2, Issue 19, p. 1250). MDPI. <https://doi.org/10.3390/proceedings2191250>
- Canché, M., & Ochoa, S. F. (2019). A Survey of Development Strategies for Collaborative Systems. In W. Shen, H. Paredes, J. Luo, & J.-P. A. Barthès (Eds.), *Proceedings of the 23rd IEEE International Conference on Computer Supported Cooperative Work in Design*, CSCWD 2019, Porto, Portugal, May 6-8, 2019 (pp. 261–266). IEEE press. <https://doi.org/10.1109/CSCWD.2019.8791934>
- Canché, M., Ochoa, S. F., Perovich, D., & Gutierrez, F. J. (2019). Analysis of notations for modeling user interaction scenarios in ubiquitous collaborative systems. *Journal of Ambient Intelligence and Humanized Computing*. Online: <https://doi.org/10.1007/s12652-019-01578-7>
- Canché, M., & Pino, J. A. (2021). Requirements Elicitation for Collaborative Systems: A Systematic Review. *Proceedings of the 24th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD'21)*, 297–304. IEEE press.
- Canché, M., Ochoa, S. F., & Perovich, D. (2022a). Identifying Interaction and Awareness Services in Mobile Collaborative Applications. *Proceedings of the 24th International Conference on Human-Computer Interaction (HCI'22)*, Gothenburg, Sweden, 26 June - 1 July, 2022, 3–16. Springer, Cham.
- Canché, M., Ochoa, S. F., & Perovich, D. (2022b). Modeling Interactions Happening in People Driven Collaborative Processes. Book chapter in G. Fortino, D. Kaber, A. Nürnberger, & D. Mendonca (Eds.), *Handbook of Human-Machine Systems*. IEEE Press/Wiley. In press.
- Canché, M., Ochoa, S. F., & Perovich, D. (2022c). Understanding the Suitability of Modeling Languages and Notations to Represent Computer-Mediated Interaction Scenarios. *Lecture Notes in Networks and Systems*. Springer, Cham. https://doi.org/10.1007/978-3-030-96293-7_5

- Lumán, A., Canché, M., Ochoa, S.F. & Perovich, D. (2022d). Modeling Interactions and Awareness Capabilities of Mobile Collaborative Systems. Accepted in the *14th International Conference on Ubiquitous and Ambient Intelligence (UCAI'22)*. Lecture Notes in Networks and Systems, Springer. Cordoba, Spain, Dec. 2022. Springer (*best HCI paper award*).

Annex K. Technical Reports

The following technical report was published as a result of this research:

- Canché, M., Ochoa, S. F., & Perovich, D. (2021). CIMoL: A Language for Modeling Interactions in People-driven Collaborative Processes. *Technical Report TR/DCC-2021-3*. Computer Science Department. University of Chile. https://www.dcc.uchile.cl/TR/2021/TR_DCC-20211115-003.pdf