

Technical Report

CIMoL: A language for modeling interactions in people-driven collaborative processes

Maximiliano Canché, Sergio F. Ochoa, Daniel Perovich

Computer Science Department, University of Chile

Abstract. Mobile computing has allowed us to conceive software systems to support mobile collaborative work in several business domains, like hospital work, emergency response, and urban maintenance. A key design aspect to model these systems is the representation of the computer-mediated interaction scenarios in which collaboration among mobile workers takes place. According to literature, some modeling languages and notations have been proposed to represent these interaction scenarios. However, given the complexity of representing large scenarios and the difficulty of involving stakeholders to validate the models, among other features, such proposals have shown limitations. In order to address them, this work presents CIMoL (*Computer-Mediated Interactions Modeling Language*), which can be used to model these interaction scenarios according to a set of capabilities these representations should have. The language can be used to support practitioners and researchers of collaborative systems to specify the computer-mediated interaction scenarios, depending on their current needs or mandatory capabilities for their representations.

1. Introduction

The use of visual models emerged decades ago, as a popular approach to represent several aspects of a software product. These models have been successfully used in the software development phases to represent systems features, and also to perform early validation of these systems mainly during the analysis and design stages.

Some of these modeling languages are focused on modeling particular systems types; for instance, the systems supporting people-driven collaborative processes (PDCPs) or unframed processes, where a workflow cannot be prescribed for such a process (Cardoso et al., 2016). In PDCPs, mobile workers perform multiple tasks in parallel, assigning their attention to the most critical or urgent activity at each moment. Thus, the participants coordinate their activities on-the-fly, considering their own work context and consequently they decide with whom and when to interact.

Considering the features of PDCPs, Canche et al., (2022) identified modeling languages and notations that can be used to specify interaction scenarios involved in such processes. That work establishes a set of capabilities that should have the models built using visual languages in order to be useful in practice. Additionally, the authors conduct a study in which existing languages and notations are evaluated by experts.

Based on the aforementioned capabilities and trying to deal with the limitations of current languages and notations, this work presents CIMoL (Computer-Mediated Interactions Modeling

Language), a modeling language that can be used to represent computer-mediated interaction scenarios involved in PDCP.

Next section briefly describes the modeling languages and notations reported in the literature, which can be used to specify computer-mediated interaction scenarios. Section 3 presents CIMoL, the proposed modelling language and Section 4 shows the software modeling tool to assist Engineers to model and validate the interaction scenarios. Section 5 discusses the improvements and limitations of this language. Finally, Section 5 presents the conclusions and future work.

2. Related work

Researchers from process engineering and computer-supported collaborative work communities have studied and proposed several notations and languages to specify these interaction scenarios. Next, we briefly introduce the most relevant ones for this modelling domain.

Case Management Model and Notation (CMMN). This is a visual modeling notation (OMG, 2016), based on the Case Handling paradigm (Aalst et al., 2005), and adapted to specify processes where the activities depend on real-time evolving circumstances. A process instance is referred to as a case, and workers in charge of a particular case can decide on how the goal of that case should be reached. A case has a design-time and run-time phase. In the first phase, business analysts are engaged in the case modeling, which includes defining tasks that are always part of predefined segments in the case model. In the run-time phase, case workers execute the plan, particularly by performing tasks as planned, while the plan may continuously evolve, since the workers can change or add discretionary tasks to the plan in run-time. Although 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, CMMN models are limited to address the dynamic nature of the people-driven collaborative processes.

BPMN Plus. This modeling notation is based on the standard BPMN, and aims to be capable of modeling PDCPs (Allah Bukhsh et al., 2019). This notation 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), *event* (occurrence of real-world event related to process), and *performer* (role that should perform an activity). Although this notation has shown to be useful and expressive for modeling specific aspects of PDCPs, it does not address various of the capabilities presented by Canche et. al. (2022). For instance, the notation is targeted to engineers and BPMN experts, consequently hindering the shared understanding between developers and stakeholders.

BPMN for Sensitive Business Process (BPMN4SBP). This language allows specifying PDCPs considering six modeling dimensions (Hassen et al., 2019). Through these specifications it is possible to represent interaction scenarios and explore their dynamic. The scenarios specification includes several types of tasks and participants, and also knowledge / information flow. Although BPMN4SBP has several useful elements to model PDCPs, it has similar limitations than BPMN Plus, since it is a notation proposed to be used for technical people.

Little-JIL and hADL. (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. hADL describes how humans interact to achieve a common goal and Little-JIL depicts processes as

hierarchies of steps. The joint usage allows modelling collaborating users and collaboration objects (e.g., messages, stream, and shared artifact). Although this proposal provides an interesting modeling capability, the joint usage of these languages increases the complexity of modeling PDCPs, and reduces the feasibility to include stakeholders in the process specification, analysis and validation.

Mobile Collaboration Modeling (MCM). This visual notation (Herskovic et al., 2019) allows specifying actors and interactions among them, in scenarios of PDCPs. These interaction scenarios are specified through a directed graph, in which the nodes represent the roles and the edges represent the interactions among them. The nodes and edges have several types that characterize the participants and the services required by them to interact. This notation allows involving stakeholders in the process specification and validation, however, it only has one abstraction level to specify the interaction scenarios. Furthermore, it does not consider mechanisms for managing the complexity or size of the models representations.

IoT Modeling. This is a proposal based on MCM, and defined to represent interactions in human-centric wireless sensor networks through an interaction graph (Monares et al., 2014). The language allows designers to model complex interactions between network nodes that can be human-based sensors, mules, witness units, regular sensors, or actuators. The arcs are stereotyped as in MCM. This notation is actually targeted at developers and it is difficult to be used by stakeholders. Likewise, it does not consider mechanisms for managing the complexity or size of the collaborative process and allows modeling only human-centric wireless sensor networks.

Collaboration graphs. These graphs are extensions of social network diagrams, adapted to specify integrated business activities (Hawryszkiewicz, 2005, 2009). In order to manage the complexity and size of the models, this notation recommends using a combination of business activities, interaction graphs, and knowledge requirements as basic constructs for specifying the interaction models. Although this proposal has several valuable insights, it is limited to represent unframed processes. Moreover, the modeling concepts considered in this notation correspond to the integrated enterprises domain; i.e., it represents a domain specific language. Similar to the previous proposals, collaboration graphs are not easy to understand by stakeholders, therefore, they are little suitable to build shared understandings between developers and stakeholders. Given the interactions representations are limited in these graphs, developers are limited to derive software requirements (particularly, interaction services) from them.

Computer-mediated Interaction Modeling Notation (CIMoN). This notation visually represents the roles played by the participants in dynamic work scenarios involved in PDCPs (Canché & Ochoa, 2018; Canché et al., 2019). 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 the mobile application. This notation allows specifying essential aspects, e.g., the interaction type (synchronous, asynchronous or both of them), messages type, and the availability of the participants to interact with others. Although CIMoN was conceived to deal with 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 ease reaching of 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.

3. Computer-Mediated Interactions Modeling Language

Computer-Mediated Interactions Modeling Language (CIMoL) was designed to model interactions among roles in scenarios involved in people-driven (or *knowledge intensive* or *unstructured processes*) collaborative processes. Since limitations found in the literature to design systems that support such processes, CIMoL was conceived considering such limitations and designed using the principles of visual languages' design from (Moody, 2009).

3.1 Language Foundations

A *Process* is the main element in CIMoL. This element represents a people-driven process, which is composed of sequential *Phases* (at least one of them). Likewise, each *Phase* can be constituted by *Work Ambits*, which are interaction scenarios that can be executed in parallel and each of them can possibly be performed in more than one *Phase*. Each *Work Ambit* can be represented in CIMoL by an interactions' model between roles of actors participating in the collaborative process.

To build each interactions' model a bottom-up strategy should be followed. Moreover, the following assumptions are considered:

- The process participants 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 the current states from their local and process contexts.
- The autonomous units manage a list of pending activities that are prioritized according to a criteria defined by each participant (i.e. urgency level, if the activity is critical, etc.). The activities priority change on the 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 can be conveyed to the autonomous units, and consequently they can interpret such information and based on that, to decide their following action.
- The autonomous units decide their workflow and the temporality of their actions according to their current context.
- The modeling notation only represents the computer-mediated interactions.

3.2 Language Metamodel

A metamodel, illustrated in Figure 1, was created to describe the abstract syntax of CIMoL. Such a metamodel presents the main concepts of modeling and the relationships among them. A UML class diagram was used to specify it.

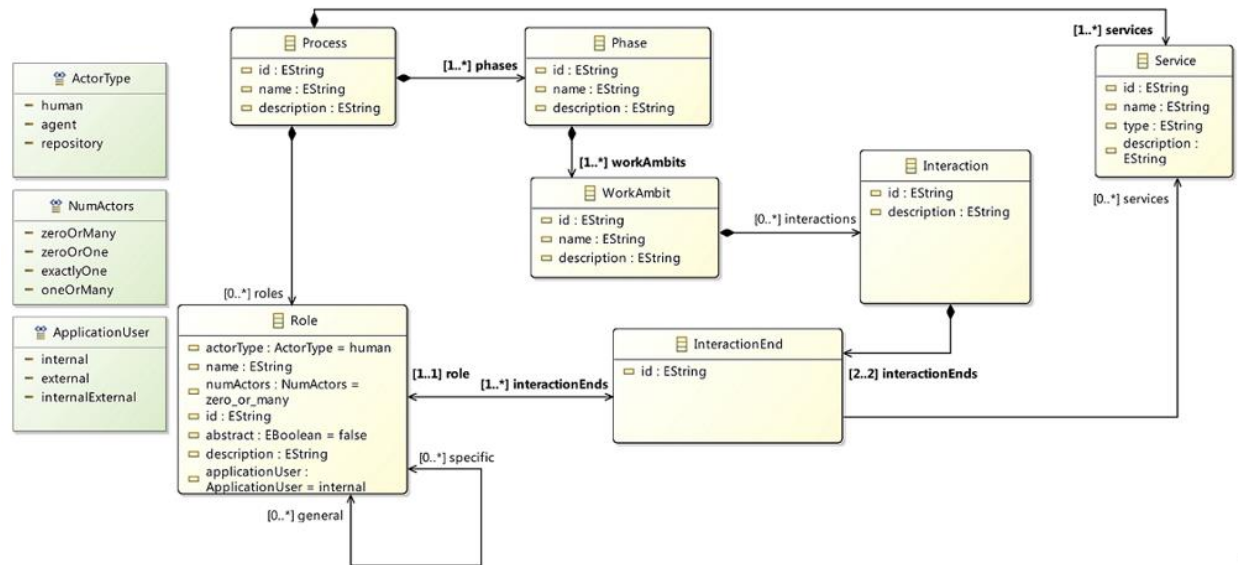


Fig. 1 CIMoL metamodel

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


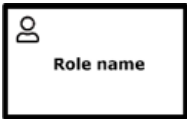




- Each *Process* contains zero or more roles (*Role* concept), which represent roles of actors participating in the process, and one or more services (*Service* concept), which represent the software services required by the roles in the process.
- Each *Process* is composed of one or more phases (*Phase* concept), which can be considered as sequential macro activities of the process.
- Each *Phase* is composed of one or more work ambits (*WorkAmbit* concept), which represent the interaction scenarios (subprocesses) we want to model. The work ambits can be executed in parallel and can belong to more than one phase.
- A *WorkAmbit* is composed of zero or more interactions (*Interaction* concept) between two roles (*Role* concept). Each interaction contains two ends (*InteractionEnd* concept), and each end is 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.

3.3 Description of the language: common constructions

The development of CIMoL considered two views (or dialects): Stakeholder view and Developer view. *Stakeholder view* is targeted to stakeholders (non-technical people involved in the elicitation stage) whereas *Developer view* is targeted to software engineers. Regardless of the dialect, each

model is based on the graph concept in which the nodes represent the roles performed by the actors participating in the collaborative process (i.e, the autonomous units), and the links indicate the interactions between them. This notation allows stakeholders and software engineers to represent each link between two nodes as a point-to-point relationship, which reduces the complexity in the design of collaborative systems. Table 1 illustrates the visual representation of the node types considered in both dialects.



Table 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 collaboration process. 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 delivered.
 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 it.

Additionally, for both dialects, the interaction between two nodes is represented as a communication link (physical or virtual) at the time they decide to collaborate. No communication link is represented if there is no interaction' requirement between two nodes. Moreover, a relationship *is-a* is used to represent that a role A can also be a role B (similarly to UML language). These two types of links are described in Table 2.

Table 2. Link types considered in the language notation

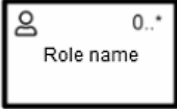
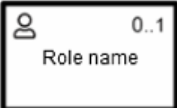
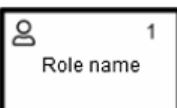
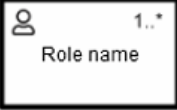
Link type	Meaning
-----------	---------

	There is an interaction between two nodes. This means that one participant requires communication with the other. A link on the same node (cycle) represents an interaction among actors playing the same role.
	There is an is-a relationship. The end node without an arrow ("son" role) can play the end role with a white arrow ("father" role). Both roles in the relationship must be of the same type (i.e. cycles are not allowed when this link is used).

3.4 CIMoL notation for the developer view

Particularly, for the developer view, the nodes' representation contains additional information in order to facilitate its analysis. In this view, both optional and mandatory roles are visually labeled, as well as the number of participants per role required to perform the process. Table 3 shows this representation. Although in the table the visual information is represented with a human-actor node type, the same representation applies to the other node types.

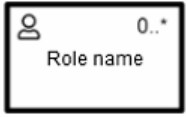
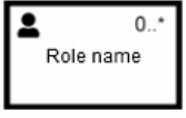
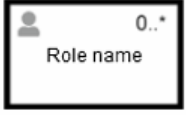
Table 3. Representation of the number of participants per role

Symbol in node	Meaning
	Zero or many. The number of role instances to conduct the process can be zero or a higher number (i.e. the role is optional).
	Zero or one. The number of role instances to conduct the process can be zero or one (i.e. the role is optional).
	Exactly one. The number of role instances to conduct the process must be exactly one, that is, the role is mandatory.
	One or many. The number of role instances to conduct the process must be at least one (i.e. the role is mandatory).

Moreover, *the user type* is visually labeled according to the application use for each node, which aims to specify whether the user will use the application in development or not. Table 4 illustrates this representation: a white-figure role represents an internal user; a black-figure role represents an external user; and a gray-figure role represents an internal/external user. Although in the table

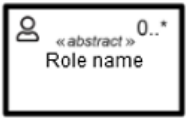
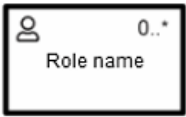
the visual information is represented with a human-actor node type and with zero-or-many instances number, the same representation applies for the other node types.

Table 4. Representation of user type according to the application use for each node

Symbol in node	Meaning
	Internal user. The users of the role will use the collaborative application in development to interact.
	External user. The users of the role will not use the collaborative application in development to interact (i.e. they possibly use an external application).
	Internal/External user: The users of the role will use the collaborative application in development, an external application, or a mix of both to interact.

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 this role will not be instantiable, that is, it will not have its own actors performing the role and must use actors of the “children” roles to carry out its activities. Table 5 shows this representation. Similar to previous specifications, although in the table the visual information is represented with a human-actor node type and with zero-or-many instances number, the same representation applies for the other node types.

Table 5. Representation of abstract and concrete roles

Symbol in node	Meaning
	Abstract role. The role does not have its own instances, that is, actors playing the role must be existing actors from the roles using the is-a relationship.
	Concrete role. The role has its own instances. By default, each role is a concrete role.

3.5 Services specification in interactions

On the other hand, regardless of the used view (stakeholder or developer view), the *communication, transmission, and interaction awareness services* must be defined for each node in order to interact with other nodes. Tables 6, 7, and 8 show the typical communication, transmission, and interaction awareness services, respectively.

Table 6. Services (requirements) supporting communication

Service	Meaning
Start audioconference	The actor playing the role can start an audio conference
Start videoconference	The actor playing the role can start a video conference
Open whiteboard	The actor playing the role can open a whiteboard
Send text message	The actor playing the role can send a text message
Send audio	The actor playing the role can send audio
Send video	The actor playing the role can send video
Send image	The actor playing the role can send images
Send structured data	The actor playing the role can send structured data
Send file	The actor playing the role can send files

Table 7. Services supporting transmission

Service	Meaning
Send to one	The actor playing the role can send data to only one receiver in the interaction
Send to many	The actor playing the role can send data to several receivers at the same time in the interaction (it includes the <i>Send to one</i> service)

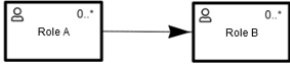

Send to all	The actor playing the role can send data to several receivers at the same time in the interaction (it includes both the <i>Send to one</i> and <i>Send to many</i> services)
Recipients can reply to sender	The recipients in the interaction can answer to the transmitter
Recipients can reply to all	The recipients in the interaction can answer to all the actors playing the sender role
Priority	The messages (data) sent by actors playing the role are labeled as priority
Encryption	The messages (data) sent by actors playing the role are encrypted

Table 8. Awareness services

Service	Meaning
Incoming messages	Awareness for incoming messages (from actors playing the other role in the interaction) is activated
Outgoing messages status	Awareness for outgoing messages is activated
Other's presence	Awareness for presence (from actors playing the other role in the interaction) is activated
Other's availability	Awareness for availability (from actors playing the other role in the interaction) is activated
Other's location	Awareness for location (from actors playing the other role in the interaction) is activated
Communication log	Awareness for communication log (history) is activated

In specific situations, specifying communication services for one role in the interaction will be required, whereas a role will be without established services (i.e. there is a unidirectional interaction regarding the services). In that case, the direction will be represented with an arrow, as shown in Table 9.

Table 9. Services directionality

Directionality	Meaning
	Role A has one or more communication services defined in the interaction, whereas Role B has not defined communication services
	This can represent two possible situations: a) both Role A and Role B have communication services defined; or b) both roles have not defined communication services in the interaction (default situation)

3.6 Interaction graph generation using CIMoL

The generation of a role interaction graph involves the steps shown in Table 10. 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 could be identified if necessary. 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 10. Steps for the role interaction graph generation with 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 (<i>communication</i> , <i>transmission</i> , and <i>interaction awareness</i> services)

The next subsection presents an application example that illustrates how to use the proposed notation, based on the model of a particular interaction scenario.

3.7 Application Example using CIMoL

In order to exemplify the use of the proposed notation, let us consider a process denominated *Informal Elderly Caregiving* in which families organize themselves to care for their older members. The example is based on the scenarios described in (Gutierrez & Ochoa, 2017), where family members assume implicit roles to fulfill their duties.

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 *Emergency Management*. Figure 2 illustrates the two representations (stakeholder and developer view) using the proposed language for the interaction scenario corresponding to the ambit *Emergency Management*. That interaction scenario involves five roles: *Emergency service*, *Caregiver*, *Elderly*, *Employee*, and *Family caregiver*. *Emergency service* is from agent type, whereas the remaining roles are human actors.

Employee and *Family caregiver*, which are specific roles, have an *is-a* relationship with *Caregiver*, which is a general role. That means the first two roles are also *Caregiver*.

Caregiver and *Elderly* have a bidirectional interaction. Furthermore, there are two unidirectional relationships: *Caregiver-Emergency service*, and *Elderly-Emergency service* (both *Caregiver* and *Elderly* can initiate communication with *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 a concrete role, actors playing such a role can have their own instances, that is, they can or cannot use instances of *Employee* or *Family caregiver* to perform their activities.

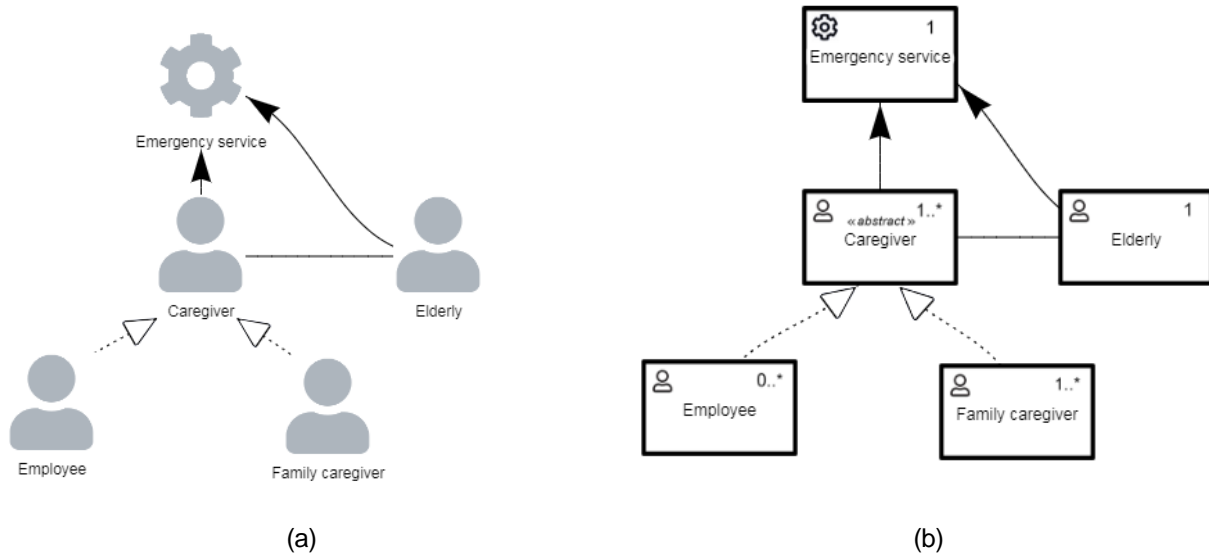


Fig. 2. CIMoL representation of the specific scenario of *Emergency Management* for the process *Informal Elderly Caregiving*: (a) stakeholder view; (b) developer view

4. CIMoL Modeling Tool

A software tool was developed to assist engineers to model and validate the interaction scenarios, considering the concepts specified previously in order to support the requirement engineering participants from the in-development application. Additionally, an automatic prototypes generator also was developed in order to help validate the requirements specified by stakeholders and engineers. The main interfaces and functionalities of this software tool are described below.

4.1 Management of processes, phases, and work ambits

4.1.1 Processes Management

The processes management's main interface is shown in Figure 3. This interface shows the existing collaborative process created and it is in charge of the management of them, which includes creating new processes, opening a selected process, as well as renaming, copying, and deleting existing processes.

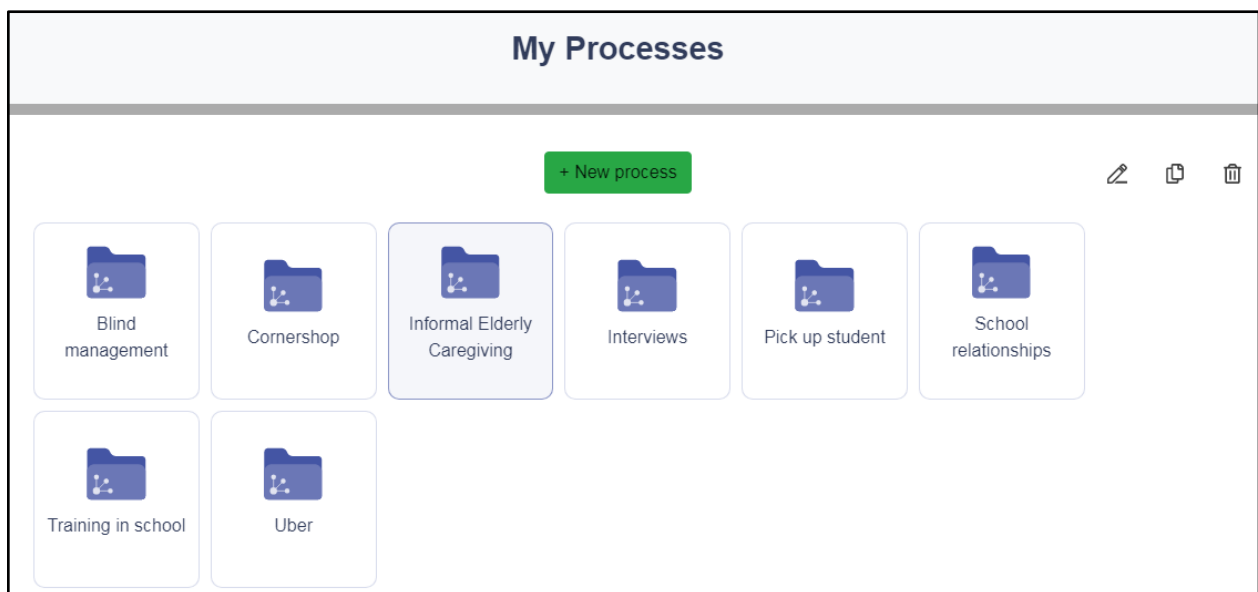


Fig. 3. Processes management's interface

4.1.2 Phases and work ambits management

When a process is edited, the interface for the management of phases and work ambits of it is visualized. For instance, the interface of the process named "Informal Elderly Caregiving" is illustrated in Figure 4. Such a process is constituted by two phases, each of which has a different number of work ambits.

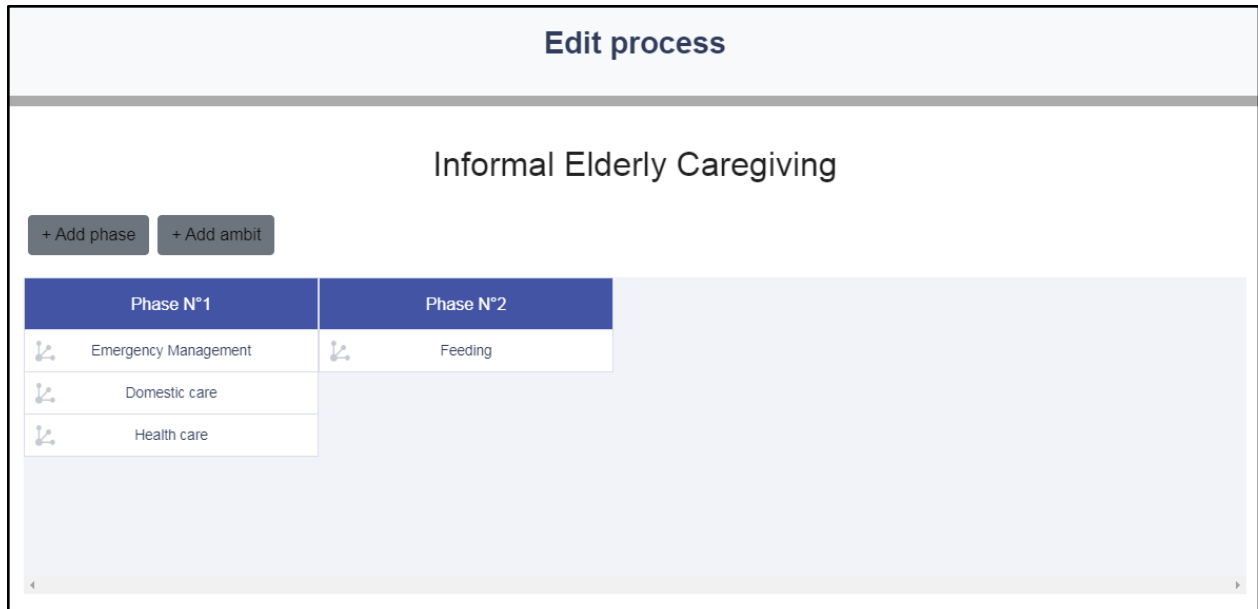


Fig. 4. Interface for the management of phases and work ambits from the process *Informal Elderly Caregiving*

The functionality of this interface includes the following:

- *Add phase*: which is performed in a sequential manner, that is, creating the next number after the last phase.
- *Rename* and *delete phase*: which are performed through a contextual menu in each phase (Figure 5).
- *Add ambit*: which allows us to assign the new ambit to one or more phases, as shown in Figure 6, which means the ambit can belong to more than one phase.

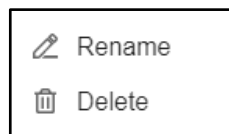
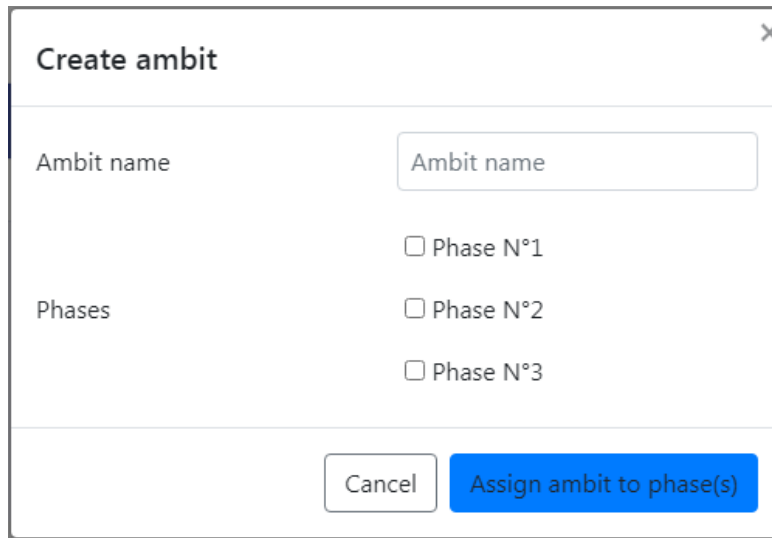


Fig. 5. Contextual menu for each phase

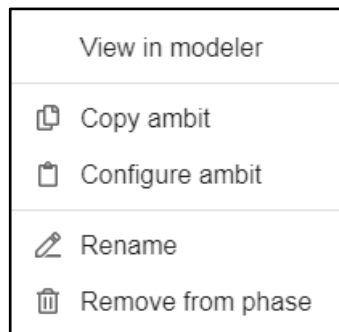


The 'Create ambit' dialog box features a title bar with a close button (X). It contains a text input field for 'Ambit name' and a section labeled 'Phases' with three checkboxes: 'Phase N°1', 'Phase N°2', and 'Phase N°3'. At the bottom, there are two buttons: 'Cancel' and 'Assign ambit to phase(s)'.

Fig. 6. Interface for the creation of work ambits

Besides, each work ambit count on the following functionalities, such as shown in the contextual menu of the Figure 7:

- *View in modeler*: which open the interaction scenario corresponding to the selected work ambit in the visual modeler
- *Copy ambit*: which copies the selected work ambit to other phase of the process
- *Configure ambit*: which reassigns the selected ambit to zero or more phases
- *Rename ambit*: which changes the name of the selected work ambit
- *Remove ambit from phase*: which removes the selected work ambit from its current phase



The contextual menu is a vertical list of options. The first option is 'View in modeler'. The next three options are 'Copy ambit', 'Configure ambit', and 'Rename', each preceded by a small icon (a document with a plus sign, a document with a checkmark, and a pencil respectively). The final option is 'Remove from phase', preceded by a trash can icon.

Fig. 7. Contextual menu for each work ambit

4.1.3 Models manipulation

When the option “*View in modeler*” (Fig.7) is selected, the software tool opens the interaction scenario corresponding to the selected work ambit in the visual modeler interface. An example of the interaction scenario's modeler interface is shown in Figure 8, which corresponds to a role interaction graph from the work ambit named *Emergency Management*.

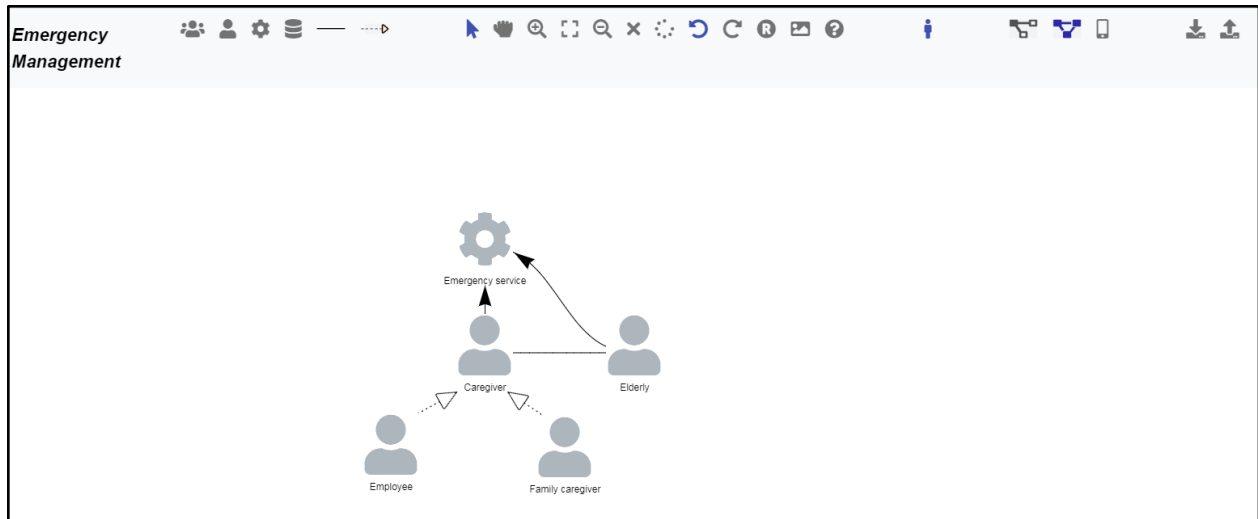


Fig. 8. Interface for role-interaction scenario's modeler (stakeholder view)

When a node is selected in this visual model interface using the *stakeholder view*, only the communication services can be established for each interaction from the node. For instance, when the node named “Elderly” is selected we can define its services (communication, transmission, and awareness services) as shown in Figure 9. For simplicity, in this figure only some services are visualized.

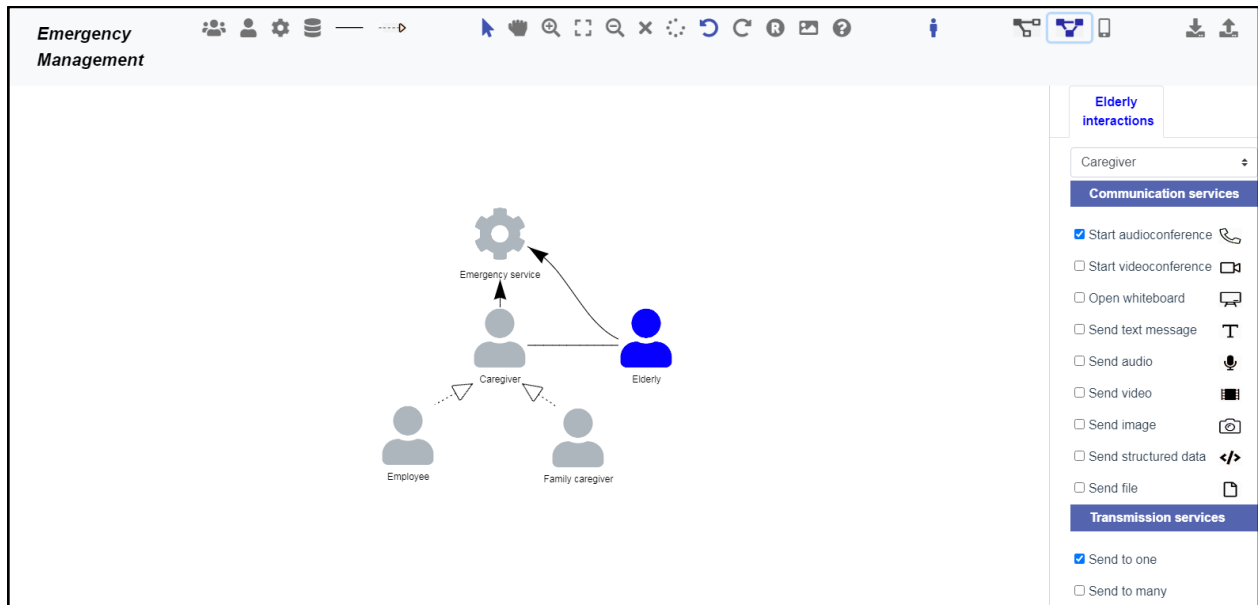


Fig. 9. Node selection in *stakeholder view*: definition of services for node interactions

Differently to the *stakeholder view*, 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 10, which shows the properties for the node named *Elderly*. With respect to abstract nodes, the properties are slightly different, as shown in Figure 11 for the *Caregiver* node. All the properties for each node type were shown in the description of the language, previous section.

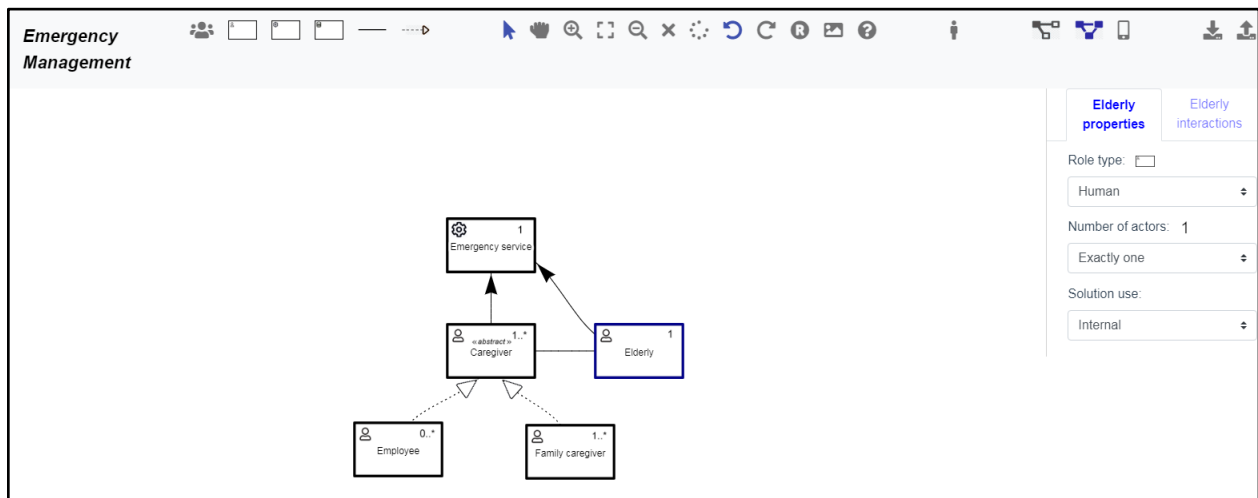


Fig. 10. Node selection in developer view: definition of properties for the node and services for node interactions

The screenshot shows the 'Caregiver properties' panel. It contains the following settings:

- Role type:** ☐ (unchecked)
- Human:** Human (selected)
- Number of actors:** 1..*
- One or many:** One or many (selected)
- Solution use:** Internal (selected)
- Abstract role:**
 - ☐ No
 - ☒ Yes
- Group members:**
 - Employee
 - Family caregiver

Fig. 11. Abstract node: *Caregiver* properties

Regarding the interactions between each pair of nodes, when one of them is selected, the services for each direction can be defined. For instance, for the Caregiver-Elderly interaction, the services for Caregiver (*Caregiver* → *Elderly*) as well as the services for Elderly (*Elderly* → *Caregiver*) can both be defined, as illustrated in Figure 12.

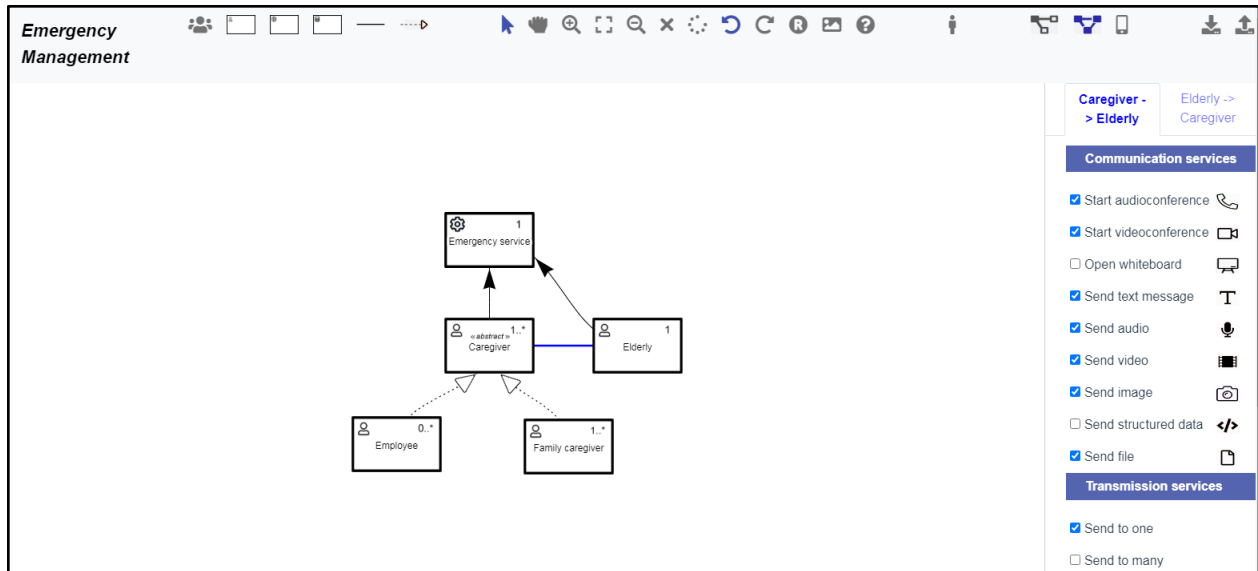


















Fig. 12. Interaction selection in developer view: definition of nodes services

Additionally, in order to support the models management, typical top-bar options were defined in the visual software tool. They are briefly described in Table 11.

Table 11. Top-bar from the visual modeler

Icon(s)	Function
	Create existing role
	Create human role
	Create agent role
	Create repository role
	Create interaction (link between two roles)
	Create an is-a relationship (relationship between two roles)
	Select element
	Panning motion

	Zoom in
	Zoom fit
	Zoom out
	Delete selected element
	Arrangement
	Undo last action
	Redo last action
	Make report (list of interaction services)
	Download graph image
	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)

4.2 Mockups of the in-development collaborative application

The software tool contains a module to generate automatic prototypes for human roles of the in-development collaborative application models in order to help the RE participants in the requirements validation. In that, software services specified in the interactions for each role are visualized in the form of visual elements. In Figure 13 the main interfaces of the mockups for the role *Caregiver* are illustrated.

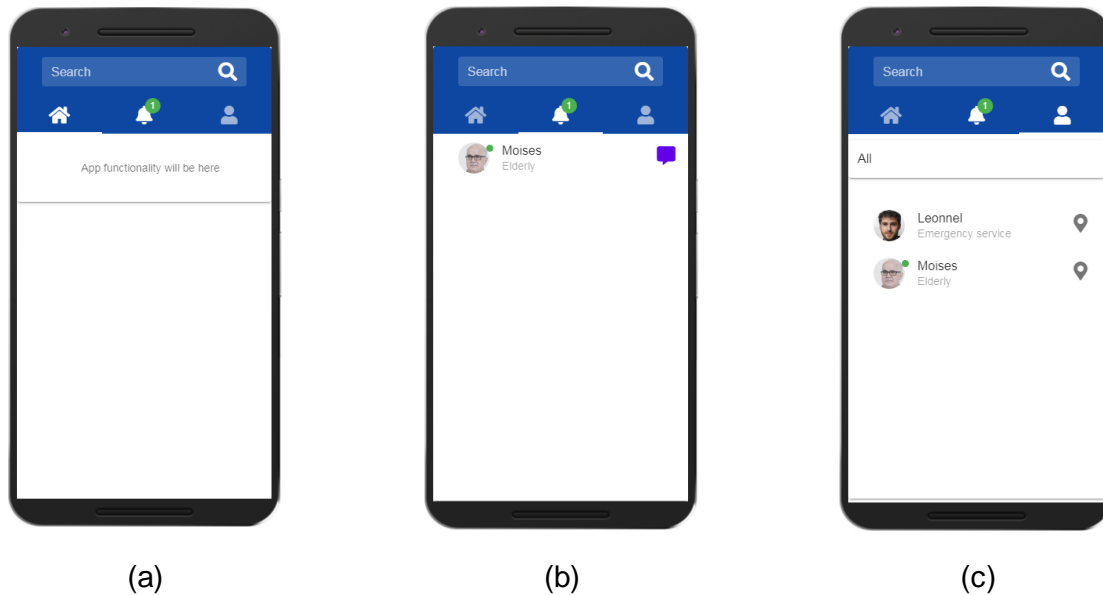


Fig. 13. Main interfaces of the mockups for the role *Caregiver*: (a) Home; (b) Notifications; (c) Contacts

Moreover, this module contains in its right size a role selector, in which the role desired can be selected in order to visualize its interactions information, as shown in Figure 14. Such information can be customized (modifying the cells values on the table) in order to show how that information is visually represented at-the-moment in the mockups.

Role:

Caregiver

Communication and awareness services:

Service name	Emergency service	Elderly
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		

Fig. 14. Interactions information of a selected role

4.2.1. Home mockup

The main insight for this mockup is to take into account the business functionality and explain to stakeholders why this space exists in the mockups. Since such a functionality is not the aim of this work, we only represent it with the label “*App functionality will be here*” (Figure 13.a). For example, if the collaborative application is targeted to *Emergency Management* issues, the functionality will be related to them.

4.2.2. Notifications mockup

This mockup shows the visual representation for the awareness service named “*Incoming messages*”, which refers to the notification of incoming messages towards the selected node. For instance, Figure 13.b shows the incoming messages for the role “Caregiver”, which include a message from an actor with the role “Elderly”. That means, the awareness service named “incoming message” for the role “Caregiver” was selected in its interaction with the role “Elderly”. Additionally, a message icon is shown on the right side of each notification on this interface in order to help understand its meaning.

4.2.3. Contacts mockups

This mockup initially shows actors playing roles that can be contacted by actors performing the role “Caregiver”. For instance, Figure 13.c shows all the actors whose roles have interaction with the role “Caregiver”, specifically with actors playing the roles “Emergency service” and “Elderly”. Additional awareness information can be displayed in the Contacts mockup: if the awareness service named “Other’s location” is selected for each role with interaction with the current role, a location icon is visualized on the right side of the Contacts mockup in order to help understand its meaning, as shown in Figure 13.c.

Besides, this mockup allows us to select a specific role in order to show only actors playing such a role. For instance, it is possible to select the role “Elderly” (Figure 15.a) in order to visualize all the actors performing that role (Figure 15.b).

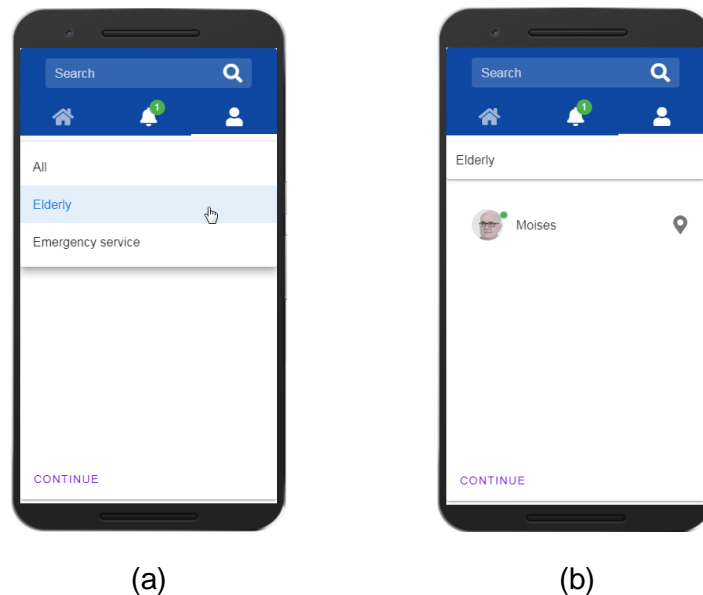


Fig. 15. Selecting a specific role in Contacts mockup

In order to help understand to stakeholders how the interaction services can be embedded in an collaborative application, different icons were integrated in these mockups as illustrated in the Figure 16 (a-d). When communication is established with an actor playing a specific role with which is carrying out interaction, a mockup with the structure of the Figure 16.a is visualized. In that, we can observe several elements representing the services specified for the interactions in the software modeling tool. Figure 16.b shows the particular mockup when a call is taken. Figure 16.c illustrates the result when the whiteboard is enabled, and Figure 16.d shows when the option of videoconference is selected. The icons in such mockups are briefly described in Table 12.

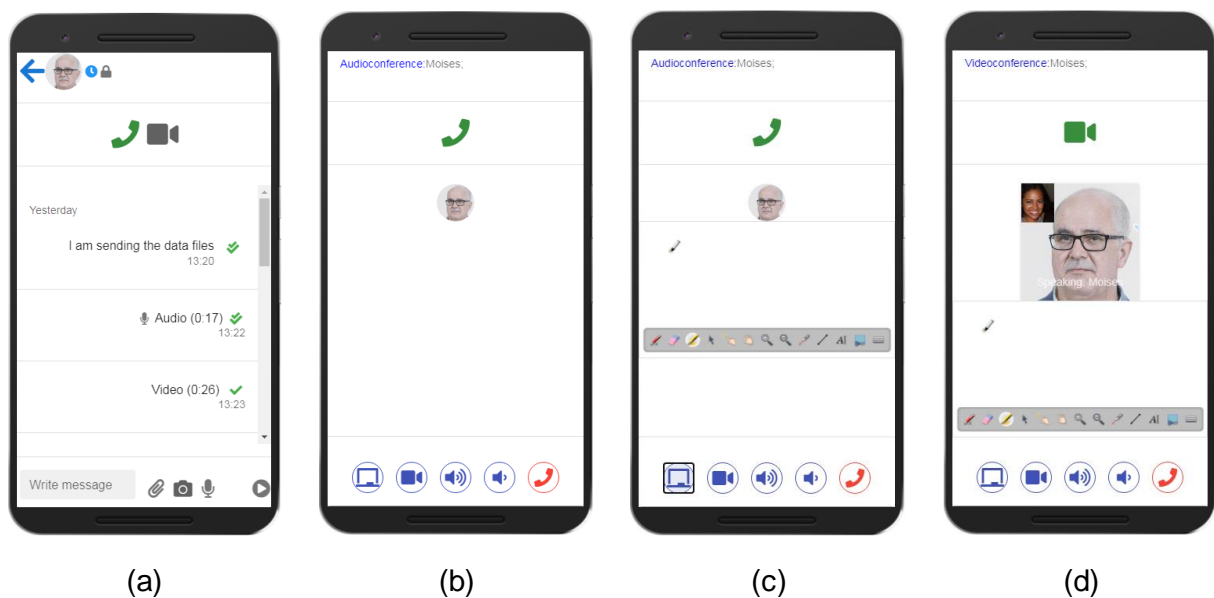














Fig. 16. Interactions services embedded in the mockup

Table 12. Icons for the Interactions services embedded in the mockup

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

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 select only one actor if the transmission service named “send to one” is selected. 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 selected.

5. Discussion

Most of the model representations make explicit the roles of the actors participating in a PDCP, and also the interactions among them. However, they still are limited to involve the stakeholders in the process of building a joint and agreed interaction schema, and then derive interaction services from it. If the interaction schemas (or scenario representations) are easy to understand only for technical people, the capability of the development team to analyze, validate and refine the interaction scenarios will be limited.

Trying to address such difficulty and other limitations of existing languages, some features of CIMoN (which is the CIMoL's predecessor) are redesigned in order to improve its capabilities. Concerning the capabilities “*representing the interactions at different abstraction levels*”, and “*managing the complexity and size of the interaction scenario specifications*”, which were described in (Canché et al., 2022) and represent limitations for several existing visual languages, they have been addressed properly by CIMoL. Additionally, regarding the capabilities “*being understandable for stakeholders*” and “*easing the building of a shared understanding between stakeholders and developers about the scenarios to be supported*”, CIMoL has defined two visual dialects targeted to two different audiences (developers and stakeholders).

Of course, researchers and practitioners can decide to overlook CIMoL features related to these capabilities. For instance, if the interaction scenarios that people have to represent are small and

with low complexity, or if the stakeholders have enough experience with visual languages, language's capabilities related to these aspects may not be used. However, it is worth knowing the advantages of the language if the use of such features is required.

We acknowledge the software modeling tool has some limitations. For instance, it does not exploit all the capabilities described in the metamodel of the language since this tool has predefined services and it does not allow to add more of them dynamically, whereas for the metamodel it is not a constraint. Another limitation is the lack of an automatic procedure to integrate the information of several sub-models towards one representing the whole system. However, developing extensions to the software tool considering such features, such limitations can be addressed.

6. Conclusions and future work

In this work, a role-interaction modeling language named CIMoL was presented. This proposal considered a set of capabilities that the models built using visual languages should have in order to be useful in practice. For instance, in order to address the capabilities “Being understandable for stakeholders” and “Easing the building of a shared understanding between stakeholders and developers about the scenarios to be supported”, the language includes two dialects: *stakeholder view*, which is aimed at customers, and *developer view*, which is targeted towards suppliers (engineers involved in the development of a collaborative system).

Although this new proposal seems to address the limitations of existing proposals, and be more usable and expressive than its predecessor, CIMoN, it will be confirmed by performing future work. Future work includes the following tasks:

- Evaluate the capabilities of CIMoL specifications to “be understandable for stakeholders” and “ease the building of a shared understanding between stakeholders and developers about the scenarios to be supported”. The models built using visual languages should reach these objectives in order to be useful in practice.
- Evaluate the concrete syntax of CIMoL.
- Implement the automatic generation of the complete model, based on a set of simple interaction models. The modeling tool will also allow to generate a complete interaction model by composing simple models (small representations) generated by the designers.

This proposal represents a modeling language to specify people-driven processes, and the user interaction scenarios involved in it. Thus, the language will help software engineers inform the analysis and design of the collaborative systems that support these processes.

References

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>
- 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. *E-Business and Telecommunications* (pp. 81-103). Springer International Publishing.
- Canché, M., & Ochoa, S. F. (2018). Modeling Computer-Mediated User Interactions in Ubiquitous Collaborative Systems. En 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, Número 19, p. 1250). MDPI.
<https://doi.org/10.3390/proceedings2191250>
- 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., Ochoa, S. F., & Perovich, D. (2022) Understanding the Suitability of Modeling Languages and Notations to Represent Computer-Mediated Interaction Scenarios. Accepted in Information Technology and Systems. *Lecture Notes in Networks and Systems*. Springer. To appear in Feb. 2022.
- Cardoso, E., Labunets, K., Dalpiaz, F., Mylopoulos, J., & Giorgini, P. (2016). Modeling Structured and Unstructured Processes: An Empirical Evaluation. En I. Comyn-Wattiau, K. Tanaka, I.-Y. Song, S. Yamamoto, & M. Saeki (Eds.), *Conceptual Modeling* (pp. 347-361). Springer International Publishing.
- 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>
- Dorn, C., Dustdar, S., & Osterweil, L. J. (2014). Specifying Flexible Human Behavior in Interaction-Intensive Process Environments. En 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. En X. S. Wang, I. Cruz, A. Delis, & G. Huang (Eds.), *Web Information Systems Engineering—WISE 2012* (pp. 143-156). Springer Berlin Heidelberg.
- 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*, 1618-1630. <https://doi.org/10.1145/2998181.2998314>

- Hassen, M. B., 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>
- 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. En 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.
- McCarthy, J. (1993). Notes on Formalizing Context. *Proceedings of the 13th International Joint Conference on Artificial Intelligence - Volume 1*, 555-560.
- Monares, A., Ochoa, S. F., Herskovic, V., Santos, R. M., & Pino, J. A. (2014). Modeling interactions in human-centric wireless sensor networks. En 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 Int. Conf. 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>
- OMG. (2016). *Case Management Model and Notation Specification*. URL: <https://www.omg.org/spec/CMMN>. Last visit: Nov 2021.