Specifying Multimodal Collaborative User Interfaces: 
A Comparison of Four Notations

Frédéric Jourde¹, Yann Laurillau¹, Alberto Moran², Laurence Nigay¹

¹ Grenoble Informatics Laboratory (LIG)
38042, Grenoble, France
² Facultad de Ciencias, UABC,
Ensenada, Mexico
alberto_moran@uabc.mx

Abstract. Interactive systems including multiple interaction devices and surfaces for supporting the collaboration of a group of co-located users are increasingly common in real applications. These include collaborative and multimodal military command posts, the latter of which is one of our application domains. Nevertheless few collaborative and multimodal interface specification notations are proposed. As a first step towards a notation for specifying a design solution prior to its software design and development, we adopt an empirical approach. In this paper we apply and compare four existing notations for collaborative systems by considering a case study, namely, a system for supporting informal co-located collaboration in hospital work. Since the selected notations differ in their descriptive qualities, with some focusing on collaborative tasks while others focus on the users’ roles and on collaborative situations, our goal is not to empirically evaluate the notations. Our goal is rather to assess their complementary aspects and their projected ability to specify a multimodal collaborative user interface.

Keywords: CSCW, multimodality, multi-devices, specification notation.

1 Introduction

The multimodal domain, including multi-surface and multi-device areas, has expanded rapidly. Significant achievements have been made in terms of both modalities and multimodal applications especially for Computer-Supported Cooperative Work such as co-located collaboration in a smart room. Due to conceptual and predictive progresses and the availability of numerous modalities, real collaborative multimodal systems are now built in various domains including medical [11,13] and military ones. As a concrete example, as part of a French military project, we study the design of command posts for controlling a group of military drones. Such command posts involve multiple distant and co-located users with various roles that are dynamic and with different interaction resources. Moving away from research prototypes, we now observe the need for specifying such systems especially in the
context of industrial projects. In this article, we address this problem of specification of multimodal collaborative User Interfaces (UI).

Specifying user interfaces is a well-established discipline and various notations have been proposed for specifying the tasks, the dialog elements, the sequences of interaction, concrete UI elements, dynamics of group behavior and so on. Such a variety of notations both in terms of their descriptive qualities, their syntactic structures and the amount of support that they offer according to the development phases has already been highlighted ten years ago in [6]. In [10], the review of notations for interaction design underlines that the most common interaction representational needs are covered by four models: task, domain, abstract and concrete UI. Many of these notations are dedicated to single user WIMP interfaces and we are interested in studying the proposed extensions of these notations and more recent notations dedicated to collaborative and multimodal UI.

As a starting point for our study, we are focusing on (1) the descriptive power of UI specification notations and especially (2) collaborative UI specification notations:

1. Descriptive power of notations for specifying UI: In [1], three dimensions for evaluating an interaction models are described: descriptive power (i.e., ability to describe a UI), evaluative power (i.e., ability to help assess multiple design alternatives) and generative power (i.e., ability to help designers create new designs). We first study existing notations along their descriptive power. Their impact on the design including their evaluative and generative powers will be studied afterwards along with tools supporting the exploitation of the notations.

2. Notations for specifying collaborative UI: Since we are interested to also model multimodal interaction, while studying collaborative UI specification notations, we also examine their power of expression for specifying concrete UI. The relationships between collaborative and multimodal interaction open a vast world of possibilities that has not been systemically explored. Going further than considering multimodal aspects such as the CARE properties [2] for the concrete UI and collaborative aspects for the abstract UI, we aim at specifying the case of tightly coupled interaction when two users are continuously engaged with the accomplishment of physical actions that are mutually observable and dependent on a time interval as well as the impacts of users’ roles for multimodal outputs.

Our approach for studying existing notations for specifying collaborative UI is empirical: we start from existing notations and we apply them for specifying a case study: a system for supporting informal co-located collaboration in hospital work. This approach does not aim at evaluating the selected notations that can be studied in light of the criteria identified in [6] and of the notational dimensions of the framework “cognitive dimensions of notations” [5]. Since the selected four notations differ in their descriptive qualities, some focusing on collaborative tasks while others on the users’ roles and on collaborative situations, our goal is to assess their complementary aspects and their projected ability to specify a multimodal collaborative user interface.

The remaining of the paper is organized as follows: first we briefly introduce our case study. We then explain the rational for selecting the notations before presenting the resulting specifications of the case study along the four selected notations. For each specification based on a notation, we present the salient aspects of the specification and conclude on the application of the notation. We finally conclude the
paper by a general discussion comparing the obtained specifications and highlighting the complementarity of the notations.

2 Case Study: Hospital Work

As highlighted by the field study described in [11], hospital medical workers including physicians and medical interns are very mobile and need to opportunistically and informally establish co-located collaboration while focusing on a particular patient. The collaborative system [11] provides a support for such informal co-located collaboration by allowing multiple users to study medical documents. In the following sections, we apply existing notations for specifying the collaborative multi-surface user interface of this system, presented in Figure 1.

![Fig. 1. Informal meeting between two physicians: Sharing and remote control of a large display from a PDA.](image1)

![Fig. 2. Graphical user interface on the PDA including a radar view of the large display (A) and control tools (B).](image2)

Using the system, two physicians can share extracts from a patient medical record such as an X-ray image and textual notes as shown in Figure 1. Such information is displayed both on the large screen (i.e., public screen) and on the PDA screen (i.e., private screen): (i) on the public screen, physicians can only annotate the medical information using a virtual pen; (ii) on the PDA, a physician can initiate a shared session, select documents to share, edit documents and stop a session. In particular all the editing actions are performed on the PDA using the stylus.

The two physicians have therefore asymmetric interaction capabilities during a meeting. The physician holding the PDA can initiate/stop a shared session. To start a shared session, the user must first run the shared meeting application. The system then automatically identifies the shareable devices in the proximity and a list of devices is displayed on the PDA screen. Once a device is selected from the list, the screen of the corresponding device (i.e., shared device) is echoed on the screen of the PDA (Figure 2). Moreover the mouse pointer of the shared device is controlled from the PDA using the stylus and the corresponding telepointer is displayed on the screen of the shared device (i.e., the public display of Figure 1). On the PDA, the private zoomed view of the public display is controlled thanks to the radar view (Figure 2-A) and control buttons (Figure 2-B).
3 Rational for the Selected Notations

Table 1 synthesizes our review of existing notations for groupware. We emphasize their descriptive qualities with the column entitled “focus” in Table 1. Except for Orchestra which focuses on work situations and their dynamics, TKS on users’ knowledge involved in task behavior, and UML-G on modeling shared data, all the other notations focus on both individual and collaborative tasks. Such notations aim at accommodating several aspects of collaborative work situations into a task specification and thus extend task specification with contextual information.

A first way for selecting the notations to be applied to our case study would be based on the syntactic structure of the notations (i.e., graphical, tabular, textual approaches) as in [7]. This solution was not satisfying since most of the notations imply several types of representations (column entitled “Number of representations” in Table 1). Moreover since our study focuses on the specification of a collaborative user interface, as a first step in our approach, we did not consider the notations that are not dedicated to interaction tasks and system behaviors, although they may be complementary to the other notations. We therefore exclude Orchestra and TKS. Moreover although CUA is focusing on individual and collaborative tasks in the context of scenarios, its main focus is on modeling the tasks for the needs of groupware evaluation. As a conclusion, we select four notations: CTT, GTA, MABTA and UML-G.

Table 1. Existing notations for groupware.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Focus</th>
<th># of models</th>
<th>Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTT [12]</td>
<td>Collaborative and individual task analysis</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>CUA [5]</td>
<td>Collaborative work situation and Collaborative/Individual task analysis</td>
<td>4</td>
<td>no</td>
</tr>
<tr>
<td>GTA [16]</td>
<td>Collaborative work situation, work dynamics and Collaborative/Individual task analysis, concrete UI</td>
<td>4</td>
<td>no</td>
</tr>
<tr>
<td>MABTA [9]</td>
<td>Collaborative work situation, Collaborative/Individual task analysis, concrete UI</td>
<td>4</td>
<td>no</td>
</tr>
<tr>
<td>Orchestra [3]</td>
<td>Collaborative work situation</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>TKS [8]</td>
<td>Cognitive model of the collaborative tasks: knowledge involved in task behavior</td>
<td>3</td>
<td>yes</td>
</tr>
</tbody>
</table>

In Figure 3, we show the involved disciplines corresponding to the four selected notations. UML-G is an extension of a standard in Software Engineering. CTT is a well established notation for task analysis in Human-Computer Interaction, while GTA and MABTA aims at extending task analysis with elements from Social Sciences (social psychology, sociology) in order to capture key elements of the nature of groupworking.

Fig. 3. The four selected notations and their related disciplines.
4 Specifications Based on the Selected Notations

By applying the four selected notations, we present the resulting specifications of the collaborative system for hospital physicians described in Section 2. To do so, we assigned one notation per author. None of the authors was an expert in one of the notations. Moreover we did not communicate with each other in order to avoid mutual influences. However, the specifications presented below are normalized to make them more readable and to provide the means to compare the notations. Due to a lack of space, the full modelisation using the four notations is available at [17]. In particular, we remove several figures and the paragraph about MABTA but the notation is discussed in section 5.

4.1 Applying CTT

Collaborative activities may be described with CTT using multiple task trees: a collaborative task tree and an individual task tree per role. A collaborative task tree contains collaborative tasks and high-level individual tasks, described by one of the multiple individual task trees. Figure 1 shows how we model our case study with CTT with a collaborative task tree and two individual task trees associated to the PDA user and Public Screen user roles.

The CTT notation considers three kinds of tasks: individual, abstract, which must be refined into a concrete task, system, mental and collaborative. A collaborative task is always considered as an abstract task that must be composed of individual tasks. Relations between tasks are defined with operators inherited from LOTOS. For example, as shown in Figure 1, we use the following operators: concurrency (||), concurrency with information passing (||[[ ]]), enabling (>>), enabling with information passing (||[ ]>>, deactivation (>[ ]), and an unary operator for iteration (*).

Hence, the task have meeting is decomposed into three sequential sub-tasks: start shared session, interact in a shared session, and stop shared session. The two first sub-tasks are associated by the enabling operator which means that the two users are able to collaborate because the shared session is started. The deactivation operator associating the two last sub-tasks means that the PDA user may end the shared session.

The task interaction in shared session is decomposed into three concurrent sub-tasks: move telepointer, interact with shared document and show. These tasks are linked with the iteration operator. In addition, the two first sub-tasks are linked with the concurrency operator with information exchange which means that the telepointer moving on the public screen is controlled by the PDA.

Finally, the task interact with shared document is decomposed into three sequential sub-tasks: share document, annotate document, and close document. It means that the PDA user may share a document with the PS user. Then, the PS user is able to annotate it. The enabling operator with information passing explicits document sharing.

The operators used to describe the collaborative task tree can be used to describe any individual task tree but a collaborative task can not appear in an individual task.
tree. However, a link is maintained between the collaborative task tree and individual task trees using abstract tasks.

The CTT notation is originally a single-user centered task tree description notation. Then, this notation has been extended to support collaborative activities. The description of collaborative activities is made at a high-level of abstraction while an individual task tree can describe concrete tasks. But, a collaborative task is no more no less than an abstract task. In addition, we observe that an individual sub-task of a collaborative task tree is not necessarily associated to a role which is a bit confusing for the designer. In addition, reading or achieving a description of a collaborative task tree using the CTT notation is heavy because we have to go to and fro repeatedly between individual and the collaborative task trees.

![Collaborative task tree](image)

**Fig. 4.** Collaborative and two individual task tree.

This notation can be used to describe concrete tasks but does not provide support to define which interaction resource or modality are available or used to accomplish a user interaction task. The only means is to use a task identifier to specify it. For example, we do so for the task *Annotate document with pen on PS*. Then, it is not possible to formally specify and describe the heterogeneity of the interaction resources. In addition, the notation does not provide any means to represent shared objects and to specify a policy for the sharing. For example, we are able to specify that the telepointer is controlled by the PDA but we are not able to specify that the telepointer can be observed by both roles.

To conclude, the CTT notation is accurate enough to describe the interactions between a role and the system but not enough between roles. In addition, this notation suffers from a lack of complementary models to describe, for example, shared objects or interaction resources.
4.2 Applying GTA

GTA (*Groupware Task Analysis*) is a method and a notation that can be used for the task analysis of collaborative activities, based on observations *in situ*, and that can be used as a specification tool for the design of groupware. GTA provides four models to describe collaborative activities: (i) a task tree model, (ii) a collaborative activity workflow model, (iii) a class diagram of objects, (iv) a model of the environment.

GTA is based on a simple ontology, which includes the following concepts: task, role, object, agent and goal.

![GTA task model of our case study.](image)

In order to model our example, as shown in Figure 5, we consider three roles: *User*, *Physician* and *System*; the role *PDA User* is just a specialization of the role *User*. Based on these roles, we consider three agents: the Physician1 agent which is a *PDA User* and a *Physician*; the Physician2 agent which is also a *Physician*; the system agent which plays the *System* part. We model objects with a class diagram (not represented here) which includes few artifacts: a PDA, a stylus and shareable devices such as the public screen.

Figure 5 shows the task tree model. We consider one task tree per interaction surface: one for the PDA and one the public screen. The first one is the *Have Meeting using the PDA* task, associated with the *PDA User* role. This task allows a PDA user to interact with the PDA and to control a shareable device. The main sub-tasks are: (i) select a device supporting collaborative sharing; (ii) control and interact with the public screen; (iii) stop a shared session. The second task tree is the *Have Meeting using the PS* task, associated with the *Physician* role. This task tree contains few sub-tasks dedicated to the interaction with the public screen such as the annotation task. As shown in Figure 5, sub-tasks are connected to a parent task with only one operator.
This lead us to introduce empty sub-tasks in the task tree in order to model a more complex task planification that corresponds to the real context of our case study. For each task and sub-tasks, we specify what are the manipulated objects, the associated roles, the pre and post conditions and the triggered tasks. We focus on the triggered tasks because this is a means to describe the dynamic part of the group activity. Indeed, based on this description, we are able to produce an activity diagram. This diagram highlights the orchestration of the activity according to each role and how the tasks are executed over time. For example, this shows the differences between the three roles and explains why we introduce the System role. In addition, we may observe that a sharing session is always started by a PDA User role. It becomes a pending task during the execution of the Identify Shareable Devices sub-task and is resumed when the latter is over. Then, the Have Meeting with PS task is triggered: this and the Interaction in shared session task are executed simultaneously.

<table>
<thead>
<tr>
<th>User actions</th>
<th>Interface actions</th>
<th>Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVETO(&lt;stylus&gt;,@&lt;x,y&gt;)</td>
<td>SHOW(&lt;Point(x,y)&gt;)</td>
<td>Draw Point(x,y) on public display</td>
</tr>
<tr>
<td>CLICK(&lt;stylus&gt;)</td>
<td>SHOW(&lt;Point(x,y)&gt;)</td>
<td>Draw Point(x,y) on public display</td>
</tr>
<tr>
<td>HOLD(&lt;stylus&gt;)</td>
<td>SHOW(&lt;Point(x',y')&gt;)</td>
<td>Draw Point(x',y') on public display</td>
</tr>
<tr>
<td>RELEASE(&lt;stylus&gt;)</td>
<td>SHOW(&lt;Point(x',y')&gt;)</td>
<td>Draw Point(x',y') on public display</td>
</tr>
</tbody>
</table>

Fig. 6. Task annotate document in NUAN.

Finally, GTA is based on NUAN to describe a concrete task. The Figure 6 shows the modelling of the concrete task Annotate with pen on PS. Hence, we have to introduce the concept of pen and we have to explicit that the annotation is produced on every shared interaction surface. In particular, we have to explicit the tight coupling between both views.

Individual and collaborative activities are described with a typical task tree. However, rules imposed by GTA on the design of a task tree, as explained above, make it more difficult to describe. At last, we are able to do it but it produces a bigger task tree because of the dummy tasks. However, it is convenient to describe the dynamic part of the activity with the concept of triggered task. Another difficulty is to define the best strategy in order to design the task tree because our application is working with multiple interaction surfaces. Then, we decide to design one task tree per surface in order to make explicit the multiplicity of interaction surfaces and the multiplicity of devices and modalities of interaction. We do it because GTA offers no means to model this. We think that the authors had considered that every user of a groupware application interacts with the same UI running on desktop computers. Maybe it is due to the fact that the notation was published back in 1996. Once again, we are able to do it but it is difficult.

In addition, GTA supports NUAN, which enables the design of a concrete task. Again, we have difficulties to describe the concrete tasks such as Annotate document with pen on PS. Indeed, the NUAN language is useful to describe interactions for WIMP interfaces but fits not very well in the case of post-WIMP interfaces, which introduce new kinds of interaction device and modality. As shown in Figure 6, we introduce the concept of pen and we try to model the mechanic of the interaction using the given NUAN operators. We can observe that the notation consider that any
input device is behaving like a mouse and that the user is manipulating a mouse pointer or cursor. The operators CLICK, HOLD and RELEASE are explicitly associated with a mouse-like device; these operators are no more usable with new interaction devices such as a tactile surface or a finger tracking. In addition, no operator is available to describe the coupling between distributed interaction surfaces. Furthermore, this notation provides no means to describe how feedback is made between devices. To do this, we have to write it explicitly as shown in the third column of the table in Figure 6.

Finally, about role distribution, we do not know how roles are managed within a task tree description: is a role, associated to a sub-task, inherited from a parent task or not? In addition, we have to introduce the System role in order to explicit the detection of shareable devices.

To conclude, GTA does not fit well for post-WIMP interfaces, for heterogeneous technology and distributed interactive surfaces. However, we find that the concept of triggered task is a good approach to associate a static and a dynamic representation of the activities.

4.4 Applying UML-G

UML is a formalism used to design a system using multiple views: the use case view describes the system from the user’s point of view; the logic view describes it from the system point of view; the implementation view describes the dependencies between software modules; the process view describes scheduled and the concurrent tasks; and, finally, the deployment view describes the topography of the system. A system can be described using UML by 13 different kinds of diagram. The most well-known are the class and sequence diagrams.

UML-G adapts UML in order to describe a groupware application using several stereotypes. The first one is the <<shared>> stereotype. If used along with an object or a relation, it means that they may be shared during a collaborative session. An object has specific properties such as {lockable}, {observable} or {distribution}.

Stereotypes <<sharedRole>>, <<sharedActor>> and <<sharedActivities>> are inherited from the <<shared>> stereotype; it defines concepts of role, actor or collaborative activity.

![Diagram](image)

**Fig. 7.** Class diagram.

As shown in Figure 7, we consider three generic objects: Telepointer, Document and Annotation classes. The Telepointer object is a graphical and shared telepointer.
controlled by the PDA and visible on the public screen. The Document object covers the medical records shared by the physicians. The Annotation object represents the stroke drawn by a physician on the whiteboard using a pen. We can observe that these objects are collaborative objects because they are tagged with the $<$share$>$ stereotype. We consider that these objects are observable and replicated on available shareable devices (PDA and public screen). Finally, we consider two roles represented by classes PDA User and Public Screen User. These classes are tagged with the $<$shareRole$>$ stereotype because these users are involved in a collaborative sharing session. Now, let us have a closer look on relations move, share or close, and annotate. These relations are also tagged with the $<$share$>$ stereotype. It means that only one PDA User can manipulate the telepointer (1 to 1 cardinality) and can manipulate remote electronic documents (1 to n cardinality). As shown in Figure 7, the PS User can also manipulate electronic documents. Finally, only the PS User is able to annotate documents as defined by the annotate relation between the Document and PS User classes. However, the class diagram shown in Figure 7 is simplified for this paper. Indeed, the final class diagram includes the classes related to the graphical elements of the user interface, such as widgets, and the classes related to the data manipulated by the system. We choose to focus mainly on shared objects used during a collaborative sharing session.

A class diagram provides only a static view on the system. We consider an activity diagram which describes how the user interactions are planned. This diagram is organised by roles, one per column. An activity is symbolised by a rounded rectangle and a name. The planification is represented by lines connecting boxes. A sharing session is started by the PDA User with the Interact in shared session activity.

Both activity and transition diagrams enabled us to describe user interaction with the system and with artifacts such as the medical records. In addition, the sequence diagram is helpful to specify how a user event triggers a software component.

In this part, we focus on the value brought by UML-G compared to UML. We do not aim at discussing what UML brings or not to design groupware.

Firstly, the new stereotypes provided by UML-G are helpful to explicit the shared and observable objects used in our case study and the different roles involved in a sharing activity. In addition, we are able to explicit the observability on manipulated objects: this is made possible using the stereotypes on relations. Furthermore, the stereotypes are useful to highlight the collaborative aspects of the interactive system. We only use two properties: observable and distribution. The other properties are not needed for our case study.

However, we find that UML-G is hard to use in order to describe the interaction dialog. Indeed, relations in a class diagram are not a very convenient way, from a human and cognitive point of view, to describe the user interactions with a shared object and between users. A real diagram would become overloaded if we had to represent every relations and, then, not usable for the design.

Secondly, the state transition diagram is helpful to describe at finer grain the user interactions with the system and the state transitions that are observable by users. Maybe this is due to the observable property which can help the designer to explicit feedback and group awareness.
To conclude, UML-G adds value to UML for the design of groupware with the new stereotypes which introduce the concepts of shared object, roles, distribution or observability.

5 Discussion

5.1 Specifying Roles

In terms of roles of the users in the working group, the four notations explicitly support their specifications. While GTA and MABTA advocate a dedicated representation to roles and relationships between users, we describe roles using UML-G by a class diagram and the involved objects for each role. Applying CTT, the approach is different since each role is described by a task tree. As opposed to UML-G which focuses on the manipulated objects per role, CTT describes the tasks per role.

As a conclusion, role should be specified in terms of tasks (activities) and in terms of objects as shared resources. For example, in our case study, we observed that we distinguished roles according to the interaction resources: PDA or public screen. This is described in the physical model of GTA that includes the physical structure of the work environment, its objects and their locations that affect the work. Finally none of the notations proposes an explicit support for describing the dynamicity of the roles during a session.

5.2 Specifying group and individual work

5.2.1 Abstract level

For describing the group work, on the one hand, CTT and MABTA advocate a dedicated representation that combines collaborative and individual tasks. Individual tasks that are present in the group work representation include the ones that take part directly in the group work coordination, such as “starting a shared session”. The group work task models in CTT and MABTA are nevertheless different. Using CTT a hierarchical representation is used for describing the group work. The leaves of the tree are elementary individual tasks that take part in the group work. In contrast to CTT, when using MABTA the representation is not hierarchical and there is only one link between the abstract tasks: the relation “influence”. This relation enables us to depict interdependencies among the various group work tasks of one or multiple users. CTT operators refine this relation for the case of temporal interdependencies only. As pointed out in [4], in addition to temporal interdependencies related to the activity level, interdependencies are related to the object level and describe the multiple participants’ access to the same set of objects. In [14], they define a set of generic mechanics of collaboration as elementary abstract tasks for such coordination.
issues surrounding how objects are assessed. Such elementary abstract tasks are
generic since they are common to a variety of social and organizational work groups.
For example one abstract task “Obtain resource” could be part of the CTT and
MABTA group work representations.

On the other hand, GTA and UML-G do not support a representation dedicated to
the group work only. Group and individual work are described together. Using GTA,
it is possible to annotate each task with the roles and the manipulated objects. As for
GTA, with UML-G there is no explicit representation of the group work: they are
implicitly described within the class diagram by different roles manipulating the same
object. Collaborative activities are further described in the UML-G activity diagram
that highlights the relationships between the individual tasks over time.

Individual tasks are described hierarchically in CTT, MABTA and GTA for each
role. MABTA refines the work group tasks into sub-tasks while maintaining the
columns for the roles and adding new individual tasks that are not related to the group
work. As explained above, GTA advocates only one representation for both group and
individual works. Links between tasks of different roles are expressed in MABTA by
the triggered tasks that can be specified for a given task. Such decoration of the task
in GTA corresponds to the “influence” relation of MABTA. As opposed to GTA and CTT, CTT do not explicitly describe the links between the tasks of different
roles. This link is deduced from the group work representation. Indeed the task trees
for the roles and the one for the group work share common tasks. In contrast to the
hierarchical refinement approach of CTT, MABTA and GTA, the activity diagram in
UML-G shows individual work and interdependencies with respect to time and roles.

Finally only GTA and UML-G enables us to represent task flows respectively in
terms of activity diagram and sequence diagram.

As a conclusion, based on our experience on the case study, the separation between
group and individual work representations seems useful because we can clearly focus
on the specification of the group tasks separately from individual tasks dedicated to a
role. However combining the two representations as in MABTA enables us to
explicitly specify the interdependencies between roles. This could be done in CTT by
adding the role linked to each task in the group work description.

5.2.2 Concrete level

CTT and MABTA advocate the same representation for abstract and concrete tasks.
For example in CTT, one leaf of the task tree for the PDA user is “annotate document
with pen on PS”. However MABTA puts a special emphasis on the graphical
interaction modality by including another description of the interface layout. The
work group description is integrated with the interface layout model.

The GTA elementary abstract tasks are described using NUAN. NUAN enables a
precise description of both users’ actions, system feedback and dialogue states.
However NUAN needs to be extended for describing multimodal interaction since it
is mainly dedicated to WIMP user interfaces.

For UML-G, concrete tasks can be described by sequence diagrams along with
state-transition diagrams. For each object, the users’ actions on it as well as its
reactions are described. Nevertheless such specification would be extremely tedious
for a complete user interface.
As a conclusion, the studied notations dedicated to collaborative user interfaces do not allow the description of multimodal interaction at the concrete level. In our case study for example, it was not possible to explicitly specify the redundancy (one of the CARE properties of multimodality [2]) of the display (PDA and public display). One future direction is to study extensions of the ICARE notation [2] for specifying the multimodal elementary group and individual tasks. Finally distinguishing abstract/concrete tasks as well as group/individual tasks allows us to identify:
- tasks that require tightly coupled multimodal interaction when two users are continuously engaged with the accomplishment of physical actions for realizing a concrete group task;
- tasks that require loosely coupled multimodal interaction when two users are performing actions along different modalities for realizing two concrete individual tasks that define an abstract group task.

6 Conclusion and Future Work

By applying four existing notations for specifying a simple groupware where two users are working on a medical image using a PDA and a public display enables us to identify some complementary aspects in the induced representations as well as some missing aspects. We underline three key issues from this empirical study:

- The distinction between group work and individual work (per role) is useful in a specification for describing at different level of detail (i.e., abstract and concrete) a collaborative user interface from its two facets, the group and the users. However a unified representation of group and individual work enables us to depict interdependencies between users with respect to time and roles. Classical hierarchical representations such as CTT are suitable for individual tasks, while group work representations need to include specific aspects of collaboration such as in MABTA where tasks are decorated with concepts from coordination theory.

- Temporal relationships between tasks for describing group work are not sufficient: Temporal interdependencies are at the activity level and interdependencies related to the object level are required for describing the multiple users’ access to the same set of objects. UML-G focusing on shared objects can be used for describing such interdependencies.

- The specification of concrete multimodal interaction as concrete tasks involves extending the selected notations dedicated to WIMP user interfaces. To do so further studies must be done on the description of tightly coupled multimodal interaction (a concrete multimodal group task corresponding to an abstract group task) and on loosely coupled multimodal interaction (concrete multimodal individual tasks corresponding to abstract individual tasks that define a composed abstract group task).

As further work, we plan to experiment on the complementary usage of the studied notations on another case study, namely a collaborative and multimodal military command post. The focus will be on extending the notations in order to depict multimodal interaction and on studying the links between the activity (task) and shared resource (object) aspects.
References

5. Green, T.: Instructions and descriptions: some cognitive aspects of programming and similar activities, AVI’00, 2000, Italy, pp. 21-28, ACM Press.