Task Modelling in Multiple Contexts of Use

Quentin Limbourg, Nathalie Souchon and Jean Vanderdonckt
Université catholique de Louvain, Institut d'Administration et de Gestion
Place des Doyens, 1 - B-1348 Louvain-la-Neuve, Belgium
{*souchon, limbourg,vanderdonckt*}*@isys.ucl.ac.be*

08 march 2002

Abstract

The context of use in which users are carrying out their interactive tasks is continuously submitted to an evolution in the user population, the computing platforms used for the tasks, and the physical environment in which users are living. This evolution process raises a need for extending traditional task modelling to support multiple contexts of use simultaneously. To address this problem, this paper first provides a formal notation of a task model that is further refined to support the variation of conditions depending on multiple contexts of use. Key concepts are then introduced to support the task modelling process so as to create a clear frontier between the Context-dependent Task Model and the Context-Independent Task Model. The Context-Partially-Independent Task Model attempts to capture subtasks shared in many contexts of use, but not all. The use of these key concepts enable designers to build a Multi-Context Task Model, notably, by factoring out common parts from Context-dependant Task Models. All these key concepts are equally denoted with the introduced formal notation. In addition, they support designers in adopting the task modelling approach of their choice in multiple contexts of use, which is so far not allowed.

1 Introduction

For many years, user interfaces (UIs) have been developed assuming that the context of use in which they work remains constant over time: the user considered to have little or no variation, interacting with the same computing platform to carry out the same task in a non-changing physical environment. Today, this assumption is no longer satisfied as we observe:

1. **A multiplicity of users**: not only types of users become more numerous (e.g., more people are willing to interact with computers, tasks previously assigned to other types of users are now devoted to new types, the user population is increasing in diversity), but also types of user are subject to many redefinitions (e.g., users do evolve over time dynamically).

2. **A proliferation of computing platforms**: existing computing platforms, like the desktop PC, are progressively enhanced with new interaction capabilities while new platforms are emerging, such as cellular phone, Personal Digital Assistant (PDA), Pocket PC, Web Appliance, or dedicated interaction devices.

3. **A continuing evolution of the physical environment**: the organizational structure may change (thus leading to moving a role from type of user to another), the office location may change (thus resulting in task reallocation), the working circumstances may change (e.g., the user moves with her computing platform from one place to another or the user is moving across computing platforms at the same place).

Existing conditions in which users carry out their interactive tasks are progressively evolving, while new conditions are appearing. Therefore, the capability of task-based UI design (i.e., with a single, all-encompassing task model) to initiate the development process and to ensure user-centered design is questioned. In other words, a task model valid for a single predefined context of use may become no longer valid for multiple, possibly largely different, contexts of use or for variations of the context of use.

The aim of this paper is to address the problem of task modelling in multiple contexts of use by augmenting the capabilities of traditional single-context task modelling to support multiple contexts of use simultaneously. The remainder of this paper is structured as follows: Section 2 situates the scope of this paper and motivates it by highlighting some shortcomings of existing approaches. Section 3 selects a well-established task model that will be subject to a formal definition of its form and properties. Section 4 introduces our detailed definition of the context of use in terms of the previously defined formal notation and provides four key concepts (i.e., the Context-Dependent Task Model, the Context-Independent Task Model, the Context-Partially-Independent Task Model, and the Residual Context-Dependent Task Model) to support an original multi-context task model. Section 5 exemplifies the above concepts on a case study in tele-medicine. Section 6 concludes the paper by reporting on the benefits of the four key concepts supporting the multi-context task modelling and suggests some future work.

2 The development process of multi-context user interfaces

To define the scope of this paper, we rely on the reference framework for plastic UIs introduced by Calvary, Coutaz & Thevenin [4]. It identifies four major levels for producing context-sensitive UIs (Fig. 1).

1. A **Concepts and Task Model** describes how a particular task can be carried out and the domain-oriented concepts related to this task.

2. An **Abstract UI** defines working spaces (or presentation units) by grouping subtasks according to various criteria (e.g., cognitive load, semantic relationships, shared concepts), a navigation scheme between the working spaces, and selects *Abstract Interaction Objects* [23] for each concept.

3. A **Concrete UI** results in a UI full description in terms of *Concrete Interaction Objects* [23] and calls to semantic functions belonging to the semantic core.

4. A **Final UI** is represented by the complete piece of code required to run/execute the UI.

In a given context of use (e.g., the first $C1$ context in Fig. 1), each non-initial level is reached from the previous one by applying *iteration* (i.e., redefinition or recomposition performed at the same level of abstraction - 'I' loop in Fig. 1) and *reification* (i.e., transformation of an abstract level into a more concrete one - 'R' top-down arrow in Fig. 1). A second context of use ($C2$ in Fig. 1) can be reached at any level thanks to a *translation* (i.e., production of a level of the same abstraction that is tailored to another context of use). The higher level a translation is applied, the wider and the richer the range of obtainable UIs can be ("the larger the plasticity domain can be" [4]). Our approach for considering multiple contexts of use consequently focuses on the 'Concepts and Task Model' level. At this level, Thevenin [21] introduced the notion of *description* to refer to as a unified way to represent various models used in UI design. In addition, three modelling activities are introduced: *corrective decoration* depicts any description modification resulting from a modelling consideration; *factoring out decoration* separates for a given description parts common to several contexts of use from uncommon parts, *generation decoration* expresses a generation directive.

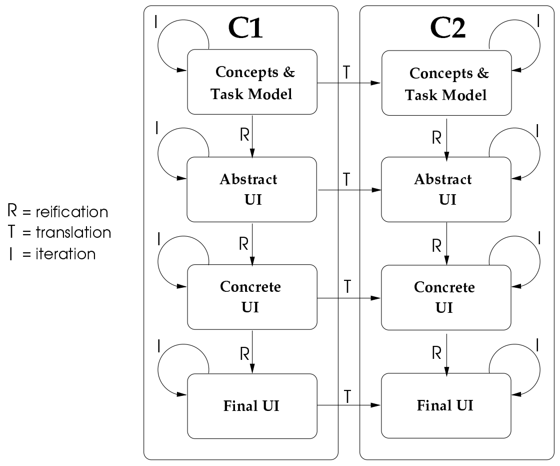


Figure 1: The reference development process for supporting context-sensitive UIs ([4]).

With respect to these three modelling activities, task modelling in multiple contexts of use can be achieved according to:

1. A *'Factoring out' approach*: build one task model for each context of use and apply factoring out decoration to separate common parts from uncommon ones. While easy to conduct, this approach can become tedious (e.g., when contexts of use are numerous), redundant (e.g., when many identical parts are considered due to the activity), unstructured (e.g., when performed with no methodological guidance), error-prone (e.g., when done by hand), or widespread (e.g., when documentation is scattered across the various models).

2. A *Minimalistic approach*: build one task model containing all parts common to all contexts of use and apply decoration for all uncommon parts resulting from specific contexts of use. While this alternative straightforwardly identifies the common parts by construction, it can be revealed hard to achieve (e.g., when numerous contexts of use or complex tasks are considered).

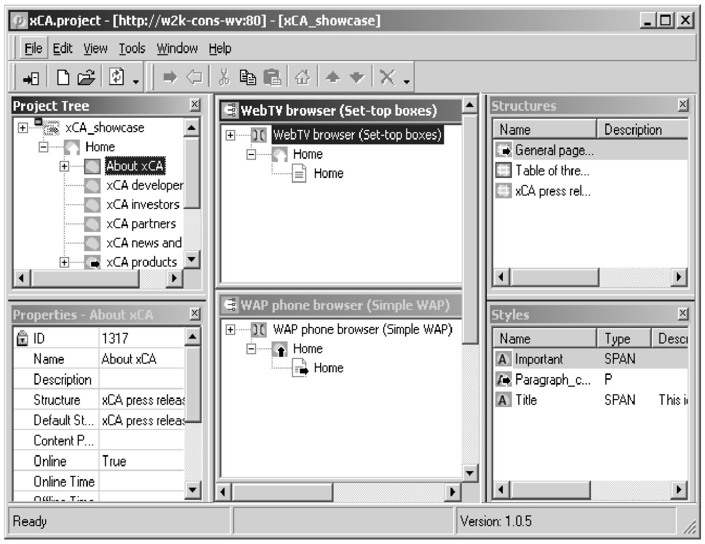
   For example, ArtStudio [4, 21] enables designers to start from a task model representing the intersection of all contexts of use and to apply corrective/factoring out decoration for multiple contexts of use. The 'One Model, Many Interfaces' developed by Paternò and Santoro [9] also relies on this approach.

3. A *Prototypicalistic approach*: build one task model for a context of use considered as representative of most cases (e.g., a more important one, a more frequent one, or a more comprehensive one) and apply corrective/factoring out decoration when appropriate. While offering a natural starting point, this approach does not specify any stopping criteria: when and where decoration should be applied is not obvious.

   For example, the polymorphic task hierarchy concept in the Unified UI Design Method [17] starts from a prototypical task model (e.g., delete a file), then apply decoration in the form of alternative task decomposition depending on the the user type (e.g., select file, then select the delete command for a blind user, select the delete command, select a file, and confirm the command for a motor-impaired user). In this case, the context of use is restricted to the user part.

4. A *Maximalistic approach*: build the most comprehensive task model with all subtasks for all contexts of use, derive from this maximal model a specific task model for each specific context of use by applying corrective/factoring out decoration. The main advantage of this approach is that the designer always relies on the same maximal model to apply decoration, thus preserving consistency. However, a shortcoming of this approach is that the quality of derived specific task models highly depends on the modelling skills of the designer. In addition, structuring subtasks in the maximal model may become complicated for sophisticated task models.

For example, the designer in xCA [1] first draws up a hierarchy of concepts and subtasks in a project tree (Fig. 2). Then, s/he drags some of these elements from the project tree and drops them into a channel tree representing a task model for a particular computing platform (e.g., a WebTV, a cellular phone mini-browser). In this case, the context of use is restricted to one component: the computing platform. Note that UIML [2] also follows a maximalistic approach without any decoration: the model is supposed to work without any variation on all intended computing platforms.


xCA.project - [http://w2k-cons-wv:80] - [xCA_showcase]
File Edit View Tools Window Help
Project Tree
xCA_showcase
Home
About xCA
xCA developer
xCA investors
xCA partners
xCA news and
xCA products
WebTV browser (Set-top boxes)
WebTV browser (Set-top boxes)
Home
Home
WAP phone browser (Simple WAP)
WAP phone browser (Simple WAP)
Home
Home
Structures
Styles
Properties - About xCA
Ready
Version: 1.0.5

ID	1317
Name	About xCA
Description	
Structure	xCA press releas
Default St...	xCA press releas
Content P...	
Online	True
Online Time	

Name	Description
General page...	
Table of thre...	
xCA press rel...	

Name	Type	Desc
Important	SPAN	
Paragraph_c...	P	
Title	SPAN	This i

Figure 2: Specific models for specific contexts of use in xCA.

The above examples show how important are the consideration of multiple contexts of use at the same time, the need for a formal notation, and an appropriate way to factoring out parts that are common to different context and for differentiating parts that are dissimilar in these context. They argue for the need of a sound basis for task modelling in multiple contexts of use.

3 Task Model

3.1 Introduction

A task model describes tasks that users need to perform in order to reach a goal when interacting with a computer-based system. Tasks are typically recursively decomposed into a hierarchy of subtasks. A task model can be represented by a graph structure where:

- **Nodes** are the different tasks and subtasks a user has to carry out.

- **Edges** denote either a decomposition relation (a task t_i is decomposed into several subtasks) or a temporal relation (e.g., a task must be performed before another) between nodes.

Task modeling has been extensively researched for years without any consensus on a formal notation. Various formalisms have been proposed (e.g., formal grammars, transition networks, Petri nets) that cover different types of information for different types of task model. Some are more oriented towards identifying the activities and their logical decomposition whereas others are including indications of temporal relationships and adding information related to various concepts such as task objects, rules or agents [14].

The selection of ConcurTaskTree (CTT) as a starting task model results from a careful analysis of several task models [13] based on the following rationale:

- CTT is more oriented towards software engineering than towards psycho-cognitive analysis (like TKS [12] for instance).
- CTT has a rich set of formally defined temporal operators (i.e. LOTOS operators) [15], probably the most extensive one.
- CTT is supported by a usable graphical tool (CCTE) which facilitates its dissemination and communication among practitioners.

This section sets the basis of a formal notation of a CTT task model in order to support task modelling for multiple contexts of use.

3.2 Definition and Properties

Let us assume that the task model is a directed graph. Let $\mathcal{RO}$ be the set of relationship operators. $\mathcal{RO}$ is partitioned into temporal and decomposition relationships. The Task Model $\mathcal{TM}$ is defined by a tuple $< \mathit{TASK}\ ,\ t_0\ ,\ T >$ where:

- TASK is a finite set, called the set of tasks. $\mathit{TASK} = \{t_0, t_1, ..., t_n\}$ where the t_i are the different tasks and subtasks that have to be carried out.
- $t_0 \in \mathit{TASK}$ is the root of the graph, that is to say the initial task.
- $T \subseteq \mathit{TASK} \times \mathcal{RO} \times \mathit{TASK}$ is a set of *transitions*, which can be noted by the triplet $< t_i, ro_i, t_j >$. As it is a directed graph, t_i is the *source node* whereas t_j is the *target node.*

For example, the task tree represented in Fig. 3 would be denoted as:

$$\mathcal{TM} =< \{t_0, t_1, t_2, t_3, t_4\}, t_0, \{(t_0, ro_1, t_1), (t_0, ro_1, t_2), (t_1, ro_2, t_2),$$

$$(t_1, ro_1, t_3), (t_1, ro_1, t_4), (t_3, ro_3, t_4)\} >$$

Moreover, some properties can be asserted:

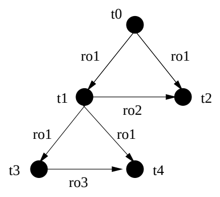


Figure 3: Example of Task Graph.

- $\forall t_i \in \mathit{TASK}$, $\Gamma^+(t_i) = \{t_j \in \mathcal{TM} \mid \exists ro_i \in \mathcal{RO} : < t_i, ro_i, t_j >\}$ denotes the set of all the successors of t_i.
- $\forall t_i \in \mathit{TASK}$, $\Gamma^-(t_i) = \{t_j \in \mathcal{TM} \mid \exists ro_i \in \mathcal{RO} : < t_j, ro_i, t_i >\}$ denotes the set of all the predecessors of t_i.
- $\forall t_i \in \mathit{TASK}$, father$(t_i)$ = set of all the predecessors of t_i where ro_i is a relationship of decomposition in the triplet $< t_j, ro_i, t_i >$.
- $\forall t_i \in \mathit{TASK}$, brother$(t_i)$ = set of all the successors or predecessors of t_i where ro_i is a temporal relationship in the triplet $< t_i, ro_i, t_j >$ or in the triplet $< t_j, ro_i, t_i >$.
- the nodes of $\mathcal{TM}$ will be organized in layers from the root. We define L_i (the layer of range i) as the set of the nodes resulting from applying Deo's level decomposition algorithm [6]. Moreover, $\forall i\ \forall j$, $L_i \subseteq \mathcal{TM}_j$ can be verified. In the above example, $L_0 = \{t_0\}$, $L_1 = \{t_1, t_2\}$ and $L_2 = \{t_3, t_4\}$.
- if $\Gamma^-(t_i) = \emptyset$, then $t_i = t_0$: the root denotes the main task.
- if $\Gamma^+(t_i) = \emptyset$, then t_i is a leaf: a leaf denotes a basic task.
- $\mathcal{TM}_i \subset \mathcal{TM}_j$ *iff* $\forall < t_i, ro_i, t_j > \in \mathcal{TM}_i \Rightarrow < t_i, ro_i, t_j > \in \mathcal{TM}_j$: $\mathcal{TM}_i$ is included $\mathcal{TM}_j$ *iff* all the transitions of $\mathcal{TM}_i$ are included in $\mathcal{TM}_j$.

For the purpose of this paper, the following hypotheses are stated:

- $\forall t_i, t_j \in \mathit{TASK}$, $\exists!\ ro_i \in \mathcal{RO} \Rightarrow < t_i, ro_i, t_j >$: $\mathcal{TM}$ is a 1-graph, that is to say that there exists only one directed edge between two nodes;
- $\mathcal{TM}$ is not a tree because $\forall t_i$, $\#\ \Gamma^-(t_i) \leq 3$: a node can have up to three predecessors : its father, its brother or itself (via iteration relationship);
- $\forall t_i \in \mathit{TASK}$, $\#\ \Gamma^+(t_i) \neq 1$: there must be more than one successor for each task, otherwise this task should not have been decomposed;
- $\forall t_i \in \Gamma^+(t_j)$: $\exists$ one brother(t_i), a corollary of the previous property;
- $\exists!\ t_i \mid \Gamma^-(t_i) = \emptyset$: there can be one and only one root for each $\mathcal{TM}$.

4 Task Model for Multiple Contexts of Use

4.1 Introduction to the Context of Use

Task models attempt to systematically represent the way users achieve a goal when interacting with a system. Some factors largely influence how a user performs tasks to achieve a goal. We group these factors under the term *context of use.*

The concept of context is extensively investigated in various areas of computer science, leading to no unique definition. Schilit *et al* [18] define context by three important aspects : *where you are, who you are* and *what resources are nearby.* It means that they include the computing environment, the user environment and, finally, the physical environment. Chen and Kotz [5] added to this definition the time context, because the moment the user has to perform a task is also an important and a natural factor.

Some authors consider context to be the user's context while others consider it to be the applications environment [20]. Petrelli *et al* [16] define the context as *any information that can be used to characterize and interpret the situations in which a user interacts with an application at a certain time.*

Dey and Abowd [7] define context to be *any information that can be used to characterize the situation of an entity, where an entity can be a person, a place or objects that is considered relevant to the interaction between a user and an application, including the user and the application themselves.* From this definition, almost any information available at the time of interaction can be interpreted as contextual information (e.g., social situation, physiological measurement, and schedules).

Schmidt *et al* [19] define context as *knowledge about the user's and IT device's state, including surroundings, situation and location.*

We define the *context of use* as the complete environment in which a task is carried out. Two types of characteristics simultaneously and univoquely determine the context of use [7, 8, 11, 10, 22, 3]:

- Characteristics that are *internal* to the system containing the application and its UI (e.g., the computing platform, the software/hardware parameters, the interaction devices, the network bandwidth, the latency, or the screen resolution).

- Characteristics that are *external* to this technical system (e.g., the type of user, her skills and knowledge, her preferences, the sound and light conditions, her geographic position in a building, the stress level, the organization structure, the information channels).

The concept of context of use is partitioned into three models:

1. The *User Model* ($\mathcal{UM}$) is a finite set $\{u_1, u_2, ..., u_n\}$ where each u_i represents a specific stereotype of user;

2. The *Platform Model* ($\mathcal{PM}$) is a finite set $\{p_1, p_2, ..., p_n\}$ where each p_i represents any property of the computing platform , such as screen resolution, operating system, or network bandwidth.

3. The *Environment Model* ($\mathcal{EM}$) is a finite set $\{e_1, e_2, ..., e_n\}$ where each e_i represents a specific configuration of physical conditions (e.g., light or pressure), location-, social and organizational environment (e.g., stress level or social interactions) in which a task is carried out;

A context C_i is denoted by a tuple $< u_i, p_i, e_i >$. A context variation appears when, at least, one element of a context tuple is modified.

A Contextual Task Model ($\mathcal{CTM}$) is defined as a task model associated with a specific context of use. A $\mathcal{CTM}$ is denoted by a tuple $< \mathit{TASK}\ ,\ t_0,\ T\ ,\ [C_{ctm}] >$, where $[C_{ctm}]$ is a matrix of context of use which holds one element: C_i.

From the example of Fig. 3, a $\mathcal{CTM}$ can be denoted as follows:

$$\mathcal{CTM} =< \{t_0, t_1, t_2, t_3, t_4\}, t_0, \{(t_0, ro_1, t_1), (t_0, ro_1, t_2), (t_1, ro_2, t_2),$$

$$(t_1, ro_1, t_3), (t_1, ro_1, t_4), (t_3, ro_3, t_4)\}, [C_1] >$$

where C_1 would be for instance: $< u_1, p_1, e_1 >$.

If an application is used in different contexts of use, a matrix $[C]$ would have more than one element of context. Some properties of an application can be asserted from its matrix of context. An application is said to be *mono-user*, respectively *multi-user* when $(\mathcal{UM}) = 1$, respectively $(\mathcal{UM}) > 1$. By analogy an application is said to be *mono/multi-environment* and *mono/multi-platform.*

4.2 Context-Independent and Context-Partially-Independent Task Model

In task modelling for multiple context of use, we notice that some tasks or subtasks are carried out the same way in all (or several) different contexts of use. Thus, isolating context-dependent tasks from context-independent ones may be considered useful.

In this section, two new concepts are defined to support this isolation: the Context-Independent Task Model ($\mathcal{CITM}$) which is a task model valid for all considered contexts of use and the Context-Partially-Independent Task Model ($\mathcal{CPITM}$) which is a task model valid for a subset of considered contexts of use. Links between different task models will be also considered.

4.2.1 The Context-Independent Task Model.

A *Context-Independent Task Model* ($\mathcal{CITM}$) integrates tasks and transitions that are common to all different contexts of use. The $\mathcal{CITM}$ is defined by a tuple $< \mathit{TASK}\ ,\ t_0,\ T\ ,\ [C_{citm}] >$, where:

- TASK is a finite set of tasks $\{t_0, t_1, ..., t_n\}$ where the t_i are tasks and subtasks that belong to each $\mathcal{CTM}$.

- $t_0 \in TASK$ is the root of the graph and of each $\mathcal{CTM}$.
- $T \subseteq TASK \times \mathcal{RO} \times TASK$ is a set of transitions common to all $\mathcal{CTM}$s.
- $[C_{citm}]$ is a matrix containing all the different contexts of use.

$$[C_{citm}] = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_n \end{pmatrix} = \begin{pmatrix} u_1 & p_1 & e_1 \\ u_2 & p_2 & e_2 \\ \vdots & \vdots & \vdots \\ u_n & p_n & e_n \end{pmatrix}$$

The following conditions must hold:

- $t_0 \in \mathcal{CITM} \Leftrightarrow t_0 \in \mathcal{CTM}_j \ \forall j$: in order to find a $\mathcal{CITM}$, all the different $\mathcal{CTM}$s need to have at least the same root. Indeed, two $\mathcal{CTM}$ having parts in common but not their root can not be considered to form a $\mathcal{CITM}$ as their main purpose is different.
- $\mathcal{CITM} \subseteq \mathcal{CTM}_i \ \forall i$ and $\forall t_i \in \{\mathcal{CITM} \setminus t_0\} \Rightarrow \exists$ father(t_i): a $\mathcal{CITM}$ is included in all $\mathcal{CTM}$s. Moreover, each task in the $\mathcal{CITM}$ (except the root) must have a father.
- $\# \ L_i$ of the $\mathcal{CITM} \geq$ *threshold* : the Context-Independent Task Model must have at least *threshold* layers. Indeed, the number of desired layers in our $\mathcal{CITM}$ should be adjustable by the designer. The relevancy of the $\mathcal{CITM}$ depends indeed on the granularity of task analysis;
- if $\Gamma^+(t_i) = \emptyset$, then t_i is a leaf task or a hinge task. A *hinge task* t_i is a task which is the source node of at least one conditional relationship with a task belonging to another task model.

4.2.2 The Context Partially Independent Task Model.

A *Context-Independent Task Model* is made up of tasks that must be carried out in **all** different contexts of use. But how do we represent a task model valid for only some of those contexts of use? For instance, if we want to develop a multi-platform application for a laptop, a desktop PC and a handheld PC, it is likely that factoring out common tasks between a laptop and a desktop PC would be useful.

A *Context-Partially-Independent Task Model* ($\mathcal{CPITM}$) integrates tasks that are valid in a subset of considered contexts of use. A $\mathcal{CPITM}$ is defined by a tuple $< TASK \ , \ t_0, \ T \ , \ [C_{cpitm}] >$, where $[C_{cpitm}]$ is a matrix containing the different contexts of use C_i with i : 1 .. m and m $\geq$ 2 and $[C_{cpitm}] \subset [C_{citm}]$.

Moreover, the following conditions must hold:

- $t_0 \in \mathcal{CPITM}_i \Leftrightarrow \exists \ t_j \in \{\mathcal{CITM}$ or $\mathcal{CPITM}_j\} \mid t_j$ is a hinge task and $\exists$ $< t_j, ro_i, t_0 >$ where ro_i is a conditional relationship;

- $\forall j$, $\mathcal{CPITM}_i \subseteq \mathcal{CTM}_j$ where $[C_{ctm}] \subset [C_{cpitm}]$ and $\forall t_i \in \{\mathcal{CPITM} \setminus t_0\}$ $\Rightarrow \exists$ father(t_i);
- if $\Gamma^+(t_i) = \emptyset$, then t_i is a leaf task or a hinge task.

We can now define more precisely a hinge task. t_i is a hinge task *iff* $\exists$ t_j $\in \{\mathcal{CITM}$ or $\mathcal{CTM}_j$ or $\mathcal{CPITM}_i\} \mid \Gamma^-(t_j) = \emptyset$ and $\exists < t_i, ro_i, t_j >$ where ro_i is a conditional relationship between two graphs ($\mathcal{CITM}$ and $\mathcal{CTM}_j$) or ($\mathcal{CITM}$ and $\mathcal{CPITM}_j$) or ($\mathcal{CPITM}_i$ and $\mathcal{CPITM}_j$) or ($\mathcal{CPITM}_i$ and $\mathcal{CTM}_j$).

4.2.3 Remark on $\mathcal{CITM}$ and $\mathcal{CPITM}$.

Two properties of the general $\mathcal{TM}$ have been relaxed in order to obtain a transient representation that shows intersection between $\mathcal{CTM}$s:

- **Unique children are allowed**. A $\mathcal{TM}$ is said to be *well-formed* iff the minimal number of children for a task is set to two. In other words, it does not make sense to decompose one task into a single task. In a $\mathcal{CITM}$ or a $\mathcal{CPITM}$, a task having only one subtask is just the sign that only one subtask is common between the different $\mathcal{CTM}$s from which the $\mathcal{CITM}$ (or $\mathcal{CPITM}$) is constructed;
- **Isolated brothers are allowed**. Each task of a well-formed $\mathcal{TM}$ has to be related at least with one of his brother. In a $\mathcal{CITM}$ (or a $\mathcal{CPITM}$), only common transitions between $\mathcal{CTM}$s are shown. As temporal relations between two brother tasks can vary from one context to another, it is admitted that two brother tasks may share no temporal relationship with each other in a $\mathcal{CITM}$ (or a $\mathcal{CPITM}$).

4.3 The Multi-Context Task Model

The Multi-Context Task Model ($\mathcal{MCTM}$) represents all possible variations of a task model for a given application. The $\mathcal{MCTM}$ components are presented in Fig. 4. A $\mathcal{MCTM}$ is the union of identified $\mathcal{CITM}$, $\mathcal{CPITM}$s and residual parts of $\mathcal{CTM}$. All components are linked with conditional relations. The residual part of $\mathcal{CTM}$s represents parts that could not be factored out in a $\mathcal{CITM}$ or $\mathcal{CPITM}$s.

A *residual* $\mathcal{CTM}$ for a context C_i is defined as the set of $t_i \in TASK$ and $< t_j, ro_k, t_l > \in T$, such that

$$\forall i,\ \forall j,\ \forall k,\ \forall l,\ t_i\ and\ < t_j, ro_k, t_l > \in\ \mathcal{CTM} \setminus (\mathcal{CITM}\ \cup\ \bigcup_i(\mathcal{CPITM}))$$

where $C_i \in [C_{cpitm}]$. A residual $\mathcal{CTM}$ can be a well-formed subgraph, a single task or a single transition.

To relate the different components of the $\mathcal{MCTM}$, a conditional expression is introduced. This condition relates a $\mathcal{CITM}$ to a $\mathcal{CPITM}$ or a residual $\mathcal{CTM}$; a $\mathcal{CPITM}$ to another $\mathcal{CPITM}$ or a residual $\mathcal{CTM}$. A condition has the form

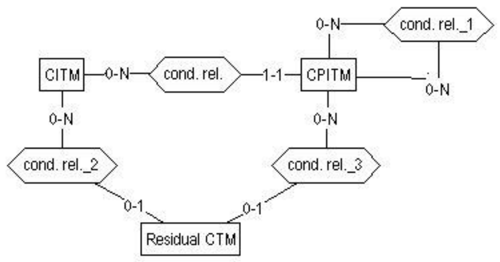


Figure 4: Multi-Context Task Model concepts.

`X/p`, where `X` specifies the contexts of use for which a subgraph is valid and `p` specifies a relationship type (decomposition or temporal) between two tasks situated on different task models (Fig. 5).

To take into account the condition, relationship type of $\mathcal{RO}$ must be subtyped into two types : simple and conditional. Four types are thus obtained: simple decomposition relationship, conditional decomposition relationship, simple temporal relationship and conditional temporal relationship.

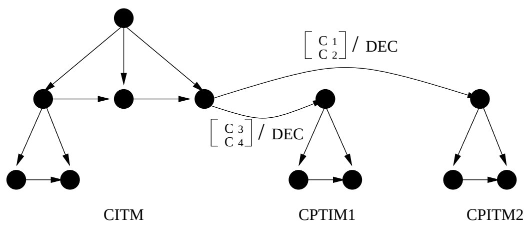


Figure 5: Conditional relations.

5 A Case Study

To illustrate how this can be applied, a case study is introduced that refines a set of scenarios taking place in a medical institution. In all scenarios, a patient is treated in an hospital and a medical staff needs to obtain all the information relative to the patient's case. Two types of person can access this information: doctors and nurses. The computing platforms on which they have to carry out their task are various: a desktop PC, a handheld PC and a Cellular Phone. Three different contexts and associated scenarios are defined:

1. **A doctor with a desktop PC (context 1)**: A doctor, in his office at the hospital, wants to prepare the visit she has to do to a patient during the afternoon. In order do this, she logs in into the system and queries a database to access the patient's medical information (Fig. 6). This information consists in medical files which are composed of text and/or

images (e.g., x-ray pictures). She may want to update this information, by adding additional observations on the patient state for instance. Moreover, for severely ill patients, the doctor also wants to monitor real-time information on the patient state (for instance, vital parameters like heart rate, body temperature).

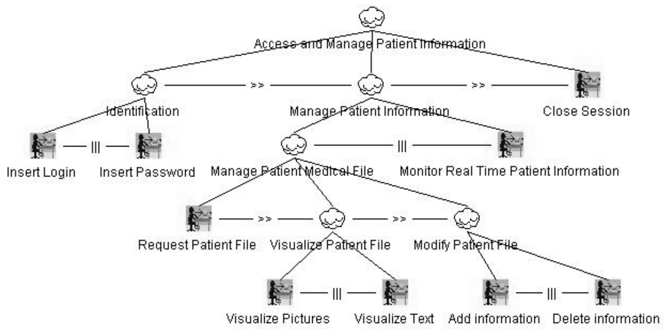


Figure 6: The $\mathcal{CTM}$ for the doctor using a desktop PC.

2. **A nurse with a handheld PC (context 2)**: A nurse is working in her service with a handheld PC. She wants to access the medical file of a patient. After logging in, the nurse queries the system to check the medical file of the patient. Considering the size of the screen of the handheld PC, the nurse can only visualize text or images one at a time. The nurse is not allowed to modify the file. Like the doctor the nurse has access to real-time parameter of a patient (Fig. 7).

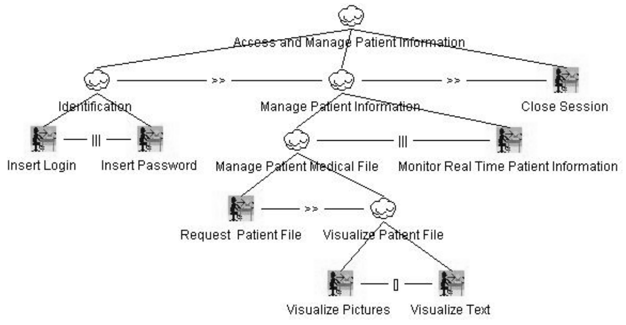
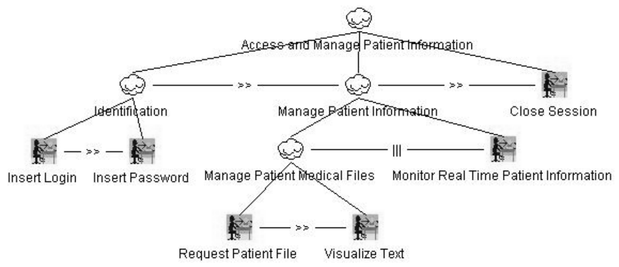


Figure 7: The $\mathcal{CTM}$ for the nurse using a handheld PC.

3. **A doctor with a Cellular Phone (context 3)**: At lunch time, the doctor wants to check a patient's medical file. After logging in into the system, she views the available textual information. As she is particularly worried about this patient, she monitors real-time information (Fig. 8).

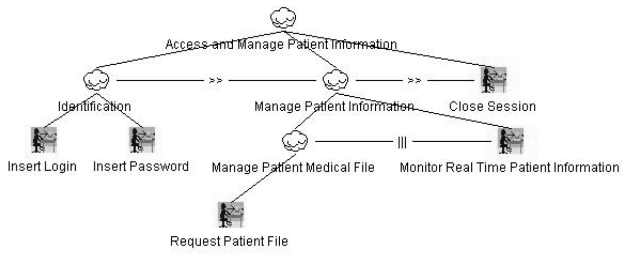


Access and Manage Patient Information
Identification
>>
Manage Patient Information
>>
Close Session
>>
Insert Login
Insert Password
Manage Patient Medical Files
|||
Monitor Real Time Patient Information
>>
Request Patient File
Visualize Text

Figure 8: The $\mathcal{CTM}$ for the doctor using a Cellular Phone.

The $\mathcal{CITM}$ (Fig. 9) is defined as : < { Access and Manage Patient Information, Identification, Insert Login, Insert Password, Manage Patient Information, Manage Patient Medical File, Request Patient File, Monitor Real Time Patient Information, Close Session }, Access and Manage Patient Information, {(Access and Manage Patient Information, dec, Identification), (Access and Manage Patient Information, dec, Manage Patient Information), (Access and Manage Patient Information, dec, Close Session), (Identification, $\gg$, Manage Patient Information), (Manage Patient Information, $\gg$, Close Session), (Identification, dec, Insert Login), (Identification, dec, Insert Password), (Manage Patient Information, dec, Manage Patient Medical File), (Manage Patient Information, dec, Monitor Real Time Patient Information), (Manage Patient Medical File, $\gg$, Monitor Real Time Patient Information), (Manage Patient Medical File, dec, Request Patient File)} > where

$$[C_{citm}] = \begin{pmatrix} Doctor & desktopPC & e_1 \\ Nurse & handheldPC & e_1 \\ Doctor & CellularPhone & e_1 \end{pmatrix}$$



Access and Manage Patient Information
Identification
>>
Manage Patient Information
>>
Close Session
Insert Login
Insert Password
Manage Patient Medical File
|||
Monitor Real Time Patient Information
Request Patient File

Figure 9: The $\mathcal{CITM}$ for the three different contexts of use.

A $\mathcal{CPITM}$ for context 1 and 2 is (Fig. 10) : < { Visualize Patient File, Visualize Pictures, Visualize Text }, Visualize Patient File, { (Visualize Patient File, dec, Visualize Pictures), (Visualize Patient File, dec, Visualize Text) } > where

$$[C_{cpitm}] = \begin{pmatrix} Doctor & desktopPC & e_1 \\ Nurse & handheldPC & e_1 \end{pmatrix}$$

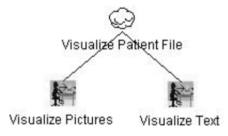


Figure 10: The $\mathcal{CPITM}$ for two contexts of use.

A $\mathcal{MCTM}$ can be defined from the different $\mathcal{CITM}$, $\mathcal{CPITM}$ and residual $\mathcal{CTM}$s (Fig. 11).

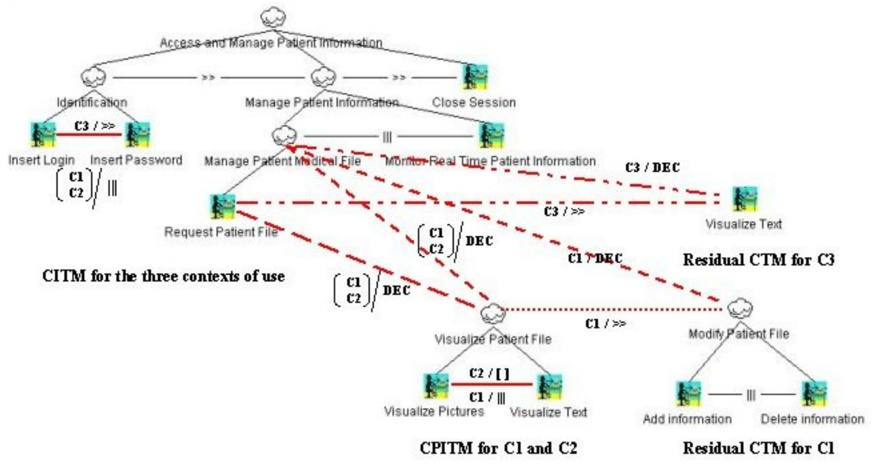


Figure 11: Multi-Context Task Model for the Case Study.

6 Conclusion and Future Work

Thanks to the approach developed in this paper, a UI intended to cover multiple contexts of use can be related to several $\mathcal{CTM}$s depending on the different contexts of use, having small or large differences depending on:

- **tasks**: (**i**) the task remains the same while the context of use changes; (**ii**) some tasks (or subtasks) are removed when the context of use changes, because either there is no possibility to perform the removed task in the new context or some tasks appear to be unnecessary or irrelevant for a certain context of use; (**iii**) task ordering is modified without modification of the tasks themselves. In this case, only the transition differ; (**iv**) some tasks (or subtasks) are added because a new context requires more tasks to achieve the same goal;
- **relationships**: the temporal relationship between two tasks may differ from one context to another. In the case study presented in Section 5, the

two subtasks of the "log in" task are concurrent in one case and sequential in another case.

In order to formally represent those possible variations, several key concepts have been defined, each of them associated with a formal notation:

- The **Context-Dependent Task Model** ($\mathcal{CTM}$) associates a task model with a context for which it applies.
- The **Context-Independent Task Model** ($\mathcal{CITM}$) represents common parts between all $\mathcal{CTM}$ of a same application;
- The **Context-Partially-Independent Task Model** ($\mathcal{CPITM}$) represents common parts between some $\mathcal{CTM}$ of a same application;
- The **Residual Context-Dependent Task Model** represents parts of the $\mathcal{CTM}$ that can not be factored out into a $\mathcal{CITM}$ or a $\mathcal{CPITM}$s;
- The **Multi-Context Task Model** ($\mathcal{MCTM}$) is a view that represents conditional relations (depending on the context variation) in the set $\{\mathcal{CITM} \cup \bigcup(\mathcal{CPITM}) \cup \text{Residual } \mathcal{CTM}\}$.

The formal notation introduced for a general task model, based on CTT, along with their use for all components of a task model for multiple contexts of use are the original contribution of this paper. They enable designers to adopt any approach discussed in Section 2 in a more formal and structured way. In particular, there is now a clear frontier between task model elements that change or do not change when the context of use is varying. The formal notation also makes it appropriate for inclusion in a tool like CTTE as it provides an internal format that can be manipulated by an automata.

With respect to the reference framework presented in Fig. 1, this work can be situated at the 'Concepts and Task Model' level and deals with the translation relationship. This study could be extended by defining a formal concept model, analyzing <task, concepts> relationships and considering the influence of context of use variations on these relationships. In particular, it could be worth to represent constraints imposed by a computing platform on the selection of presentation elements (e.g., availability vs unavailability) or preferred by a user type. Furthermore, there is a need for a formal abstract UI model and concrete UI model that could in their turn be subject to a study on context of use variation. In addition, some patterns should be identified to represent the translation relationship in prototypical context variation. Finally, the notation developed here should be extended to represent run-time adaption mechanisms, as run-time subtask switching, branching, or migrating.

References

[1] The xCA suite, 2002. accessible at http://www.x-ca.com/xca/Products/xCA.suite/.

[2] M. Abrams, C. Phanouriou, A.L. Batongbacal, S. Williams, and J. Shuster. UIML: An Appliance-Independent XML User Interface Language. In A. Mendelzon, editor, *Proceedings of 8th International World-Wide Web Conference WWW'8 (Toronto, May 11-14, 1999)*, Amsterdam, 1999. Elsevier Science Publishers. Accessible at http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html.

[3] P.J. Brown, J.D. Bovey, and X. Chen. Context-Aware Applications: From the Laboratory to the Marketplace. *IEEE Personal Communications, 4(5)*, pages 58–64, 1997.

[4] G. Calvary, J. Coutaz, and D. Thevenin. Supporting Context Changes for Plastic User Interfaces : A Process and a Mechanism. In A. Blandford, J. Vanderdonckt, and P. Gray, editors, *Joint Proceedings of HCI'2001 and IHM'2001 (Lille,10-14 September 2001)*, pages 349–363, London, 2001. Springer-Verlag.

[5] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.

[6] N. Deo. *Graph Theory with Applications to Engineering and Computer Sciences.* Prentice-Hall, Englewood-Cliffs, 1974.

[7] A.K. Dey and G.D Abowd. Toward a better understanding of context and context-awareness. Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology, 1999.

[8] J. Eisenstein, J. Vanderdonckt, and A. Puerta. Applying model-based techniques to the development of UIs for mobile computers. In *Proceedings of ACM Conference on Intelligent User Interfaces IUI'2001 (Albuquerque, January 11-13, 2001)*, pages 69–76, New York, 2001. ACM Press.

[9] Paternò. F and Santoro. C. One model, many interfaces. In Ch Kolski and J. Vanderdonckt (Eds.), editors, *Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces CADUI'2002 (Valenciennes, 15-17 May 2002)*, pages 143–154, Dordrecht, 2002. Kluwer Academics Publishers.

[10] J. Gwizdka. What's in the context. In *Proc. Of CHI'2000 Workshop on Context Awareness (The Hague, April 1-6, 2000)*, Atlanta, 2000. GVU Center, Georgia University of Technology, Research report 2000-18e.

[11] P. Johnson. *Human-Computer Interaction: Psychology, Task Analysis and Software Engineering.* McGraw-Hill, London, 1992.

[12] P. Johnson, P. Markopoulos, and H. Johnson. Task knowledge structures: A specification of user task models and interaction dialogues. In *Proceedings of Task Analysis in Human-Computer Interaction, 11th Int. Workshop on Informatics and Psychology (Schraeding, June 9-11)*, 1992.

[13] Q. Limbourg, C. Pribeanu, and J. Vanderdonckt. Towards uniformed of task models in a model-based approach. In C. Johnson, editor, *Proceedings of the 8th International Workshop on Design, Specification and Verification of Interactive Systems Workshop DSV-IS'2001 (Glasgow, June 13-15, 2001)*, volume 2220 of *Lecture Notes in Computer Science*, pages 164–182, Berlin, 2001. Springer-Verlag. to be published.

[14] F. Paternò. *Model Based Design and Evaluation of Interactive Applications.* Springer-Verlag, Berlin, 1999.

[15] F. Paternò, C. Mancini, and S. Meniconi. ConcurTaskTree: A diagrammatic notation for specifying task models. In S. Howard, J. Hammond, and G. Lindgaard, editors, *Proceedings of IFIP TC 13 International Conference on Human-Computer Interaction Interact'97 (Sydney, July 14-18, 1997)*, pages 362–369, Boston, 1997. Kluwer Academic Publishers.

[16] D. Petrelli, E. Not, C. Strapparava, O. Stock, and M. Zancanaro. Modeling context is like taking pictures. In *Proc. Of CHI'2000 Workshop on Context Awareness (The Hague, April 1-6, 2000)*, Atlanta, 2000. GVU Center, Georgia University of Technology, Research report 2000-18e.

[17] A. Savidis, D. Akoumianakis, and C. Stephanidis. *The Unified User Interface Design Method*, chapter 21, pages 417–440. Lawrence Erlbaum Associates, Mahwah, 2001.

[18] B. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications WMCSA'94 (Santa Cruz, December 1994)*, pages 85–90, Los Alamitos, December 1994. IEEE Computer Society Press.

[19] A. Schmidt, K. Asante Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde. Advanced interaction in context. In *Proceedings of First International Symposium on Handheld and Ubiquitous Computing HUC'99 (Karlsruhe, 27-29 September 1999)*, pages 89–101. Springer-Verlag, 1999.

[20] A. Schmidt, M. Beigl, and H.W. Gellersen. There is more to context than location. In *Workshop on Interactive Applications of Mobile Computing IMC'98 (1998)*, 1998.

[21] D. Thevenin. *Adaptation En Interaction Homme-Machine : Le Cas de la Plascticité.* PhD thesis, Université Joseph Fourier, 21 December 2001.

[22] G. Tsibidis, T.N. Arvantitis, and C. Baber. CHI 2000 proposal for the what, who, where, when, why and how of context-awareness. In *Proc. Of CHI2000 Workshop on Context Awareness (The Hague, April 1-6, 2000)*, Atlanta, 2000. GVU Center, Georgia University of Technology, Research report 2000-18e.

[23] J. Vanderdonckt and F. Bodart. Encapsulating knowledge for intelligent automatic interaction objects selection. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, and T. White, editors, *Proceedings of the ACM Conference on Human Factors in Computing Systems InterCHI'93 (Amsterdam, 24-29 April 1993)*, pages 424–429, New York, 1993. ACM Press.