Model-Based Design of Context-Sensitive Interactive Applications: a Discussion of Notations

Jan Van den Bergh
Expertise Centre for Digital Media
Limburgs Universitair Centrum
Universitaire Campus
B-3590 Diepenbeek, Belgium
Jan.VandenBergh@luc.ac.be

Karin Coninx
Expertise Centre for Digital Media
Limburgs Universitair Centrum
Universitaire Campus
B-3590 Diepenbeek, Belgium
Karin.Coninx@luc.ac.be

ABSTRACT
Model-based design of user interfaces can be a viable alternative for other user interface specifications especially in the case of multi-platform and even more so in the case of context-sensitive interactive applications. In this paper we look at several notations used in model-based design methodologies and analyse them according to requirements we determined for the notations. In order to get an overview of how well different approaches are able to support the design context-sensitive interactive applications, we determine a set of models that are relevant for this type of applications and organise them visually in such a way that the level of support for different models and the relations between them could be shown adaquately. The resulting information is sequentially used to determine areas where work is needed to design better notations for the involved models using a review of several notations used for model-based design of user interface or interactive systems.

Author Keywords
Model-based user interface design, context-sensitive interactive systems, graphical notation

ACM Classification Keywords
D.2.1. [**Requirements/Specifications**]: Languages, D.2.2 [**Design Tools and Techniques**]: User Interfaces, H.5.2 [**User Interfaces**]: Theory and methods

INTRODUCTION
The ever increasing diversity of devices and users and the parallel evolution of more personalisation and more general, the adaptation to context has renewed the attention given to model-based design of user interfaces. This evolution, however, has not produced many broadly accepted models and notations. This in contrast to the world of system design where the Unified Modeling Language (UML [12]) has emerged as the modeling notation of choice. There also does not seem to be an accepted method for integration of user interface modeling with system modeling.

Several notations for model-based design of user interfaces, and more generally interactive systems, have been proposed. They all use several models to support the design process. The names, contents and number of these models vary greatly. Despite of this, several models seem to come back in most approaches.

In this paper, we determine a set of requirements for a notation for model-based design of context-sensitive user interfaces. Some requirements cover the qualitative aspects of a possible notation. This because the use of a notation is highly dependent, not only on the supported semantics, but also on its usability – how it represents the information in the model as noted by Paterno in [19]. Other requirements handle the information that should be representable for the design of context-sensitive models. These requirements are reflected in a visual representation of the ideal configuration of models.

To get a better overview how well current model-based techniques are fit to design context-sensitive user interfaces, and more generally context-sensitive interactive applications, we compare the presented requirements with the coverage of models of some recent and some well known notations. The results of this comparison are than used to determine the areas where some important work still needs to be done regarding the notation of the models and where incremental work should suffice.

QUALITATIVE REQUIREMENTS
This section discusses the qualitative requirements we set for a model representation for the design of user interfaces:

understandable The notation should be relatively easy to understand and should be perceivable by as many people as possible. It is very difficult to evaluate how "understandable" a notation is. We will split this requirement is two parts: being build on a*proven basis* and having a publicly available specification.

divide and conquer In graphical notations, different aspects of the specification should be the focus of different diagrams. An abstract user interface description has different aspects that are difficult to combine understandably in a single diagram. Different diagrams, using (partially) the same symbols can be easier to understand than a single model that shows everything. Different diagrams can also be used to clarify some information that is already present in another diagram. For textual notations, a clear separation of the information of the different models is also an advantage.

not isolated The different models represent different aspects of a user interface but the information they describe is still dependent on the relations with other models; unlinked information is unused information and should not be described.

expressive The requirement for the notation to be understandable should not reduce the capacity of the notation to express complex models and/or relationships. Complex behaviour that is not relevant to the model should not be represented as such, but as simple as possible without loosing too much information.

tool support The notation should be supported by a tool. A tool can provide the necessary constraints that ensure that the information presented in possibly different diagrams are consistent. A tool can also allow the designer to hide certain information in a diagram can be temporarily hidden when desired.

MODEL REQUIREMENTS

Besides the notational requirements, model requirements — which models can be represented using the notation — are at least as important. The kind of models used in different approaches varies, however a useful categorisation is presented in [8]:

Application Model This model is also called concepts model[2] or domain model[17, 19]. In general a user interface is used to present or modify data. This data has a certain type and a certain structure that can heavily influence the way the different parts of the user interface are presented. The domain model is usually highly related to the task model, since most tasks involve the presentation or manipulation of data. The domain model is also used to bind the user interface to the functional core of the application.

Abstract Presentation Model The abstract presentation model describes the organisation of the user interface. It describes the structural aspect of the user interface; the way different interaction components are organized in a single presentational unit that can be graphical or speech-based. This model provides modality and toolkit independent information.

Concrete Presentation Model The concrete presentation model translates the information described in the abstract presentation model to concrete instantiations for a specific modality.

Task/Dialog Model The task model and the dialogue model are two separate models at different levels of abstractions and useful in different stages of analysis and design. They largely cover the same information: the organisation of the user's interaction with the application. While the task model concentrates on the tasks/activities that need to be performed, the dialogue model concentrates more on how these activities are organized in the user interface; which set of tasks is available at a certain moment in time and thus form a dialogue and how dialogues are interconnected.

For the construction of context-sensitive user interfaces these models do not describe all the necessary information, therefore other information is still needed. Calvary et al. [2] proposed a reference framework for plastic user interfaces; user interfaces that can adapt to the context in which they are used while maintaining usability. The reference framework is used to show how and when certain conceptual models (concepts, tasks, interactors, environment, user and platform) are used in the development cycle that consists of artifacts at different levels of abstraction. From abstract to concrete these artifacts are: task and concepts, abstract user interface (platform independent description of interface), concrete user interface (platform dependent description of inferface) and final user interface (interface rendered at application runtime).

The user, environment and platform model, presented in the reference framework are grouped, as they all describe information periferal to the interaction of a user with an application, into the *context model*. We believe that inclusion of services as in the environment should also be included in the context model [20]; their availability or absence can have serious effects on the user interface of an application. The description of services is also significantly different from other entities in the environment and can therefore be considered as a separate first-class element in the context description.

Besides the models themselves, the links between the models are also important. Both are shown in figure 1. Black lines denote the separation of the models as well as possible links between the models they separate. The following important possible relations can be observed in the figure:

- The abstract presentation (APM) is independent of the context (CM) and thus has no link with the context. This position is also taken by other approaches. The abstract user interface descriptions generated by the TERESA-tool [16] do not contain context dependent features, although the abstract presentation can be different for certain contexts. This difference is cause by another model; the task model that is dependent on the context.

  The APM does have relations with the concrete presentation model (CPM), the domain model (DM), required to generate user interfacesthat can take advantage of the characteristics of a certain platform and to relate the user interface to the functional core of the application.

- The task and dialog model (TDM) have relations with the context; some tasks are only relevant in certain contexts and other tasks are carried out differently in different contexts. The choice for relations between these two models is supported by Coutaz and Rey [6] and Dey [10] who state that the context is relevant for the interaction of a

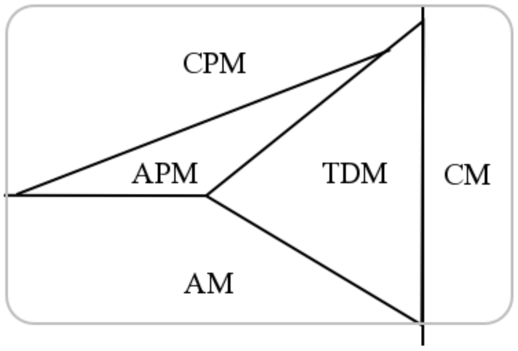

Figure 1: Reference for models in design of context-sensitive interactive applications and a proposed configuration of notations

user with the system. The model that defines this interaction is the TDM.

The task model also has relations with the domain model — tasks manipulate certain objects/artifacts — and the abstract presentation model — the user needs some interface to interact with the computing system.

- The concrete presentation model is linked with the context since the concrete presentation can depend on the platform (e.g. mobile phone or desktop system), the user (e.g. a beginner or an expert in a certain domain) or even the environment (e.g. sound alerts can be disabled when the user is a meeting).

The gray areas in figure 1 denote the graphical representations of the involved models, while the gray outline shows the wanted coverage of a textual notation and the supporting tool(s). An important thing to note is that the graphical representation should cover most but not all of the data in a model instance.

The next section will discuss the notations of different model-based design environments and methodologies and which models they cover to what extend using the graphical representation used in figure 1. Both graphical notations and textual notations are discussed.

MODELS IN EXISTING APPROACHES

Figure 2 gives an overview of which models and which relationships between those models are supported by the various approaches. The coverage of graphical notations is shown by gray regions while the coverage of textual notations is shown by gray outlines. The discussion of graphical and textual representations will be interleaved to better show differences between the textual and the graphical notations. The different approaches support the context model to different degrees and thus table 1 is provided to better understand the differences. A discussion of the qualitative properties of the notations is provided in table 2 with extra explanation in the text, where needed.

One of the most well-known graphical notations of model-based design is the ConcurTaskTrees (CTT) notation of Patero [18]. The notation is a pure hierarchical task notation that also reflects the temporal relations between tasks at the same level in the same subtree. The related XML notation, using TERESA XML, however allows much more information to be specified (see figure 2(a)). This information can include objects that are manipulated, objects that are used to interact with the application including the platforms on which they may be available. TERESA XML also supports the expression of an abstract user interface, which can be (semi-)automatically derived from the task model and associated information. The abstract user interface (AUI) description supports more or less the same reference models as the task notation, however, it is specific for a certain platform and thus does not include platform information. The TERESA tool [16] supports editing of the CTT-model and can optionally show only tasks that are relevant for a certain platform. The AUI-model is (semi-)automatically generated from CTT and minimal editing is supported. The support for the domain model is also minimal.

Clerckx et al. [5] proposed a variant on the CTT with explicit representation of the context-sensitive parts using decision nodes based on the ideas expressed in [21], resulting in a presentation that has limited support for context information. The graphical notation is supported by the tool DynaMo-AID [4] that has richer support for models: it allows the designer to precisely define in which context which tasks can be carried out using a decision tree that is linked with the decision nodes in the graphical presentation. The abstract presentation model as well as a limited description of the domain objects are also supported as is shown in figure 2(b). DynaMo-AID enables to generate a concrete prototype of the modeled application using a variety of back-ends. During the execution of the prototype, the context described in the models can be manipulated. No details are available on the exact notation used for the textual description of the model.

In earlier work [28] we also presented the Contextual ConcurTaskTrees, an extension to the CTT where the context is no longer considered to be some target, but can also be an active part of the interaction. This is realised by the introduction of the *context task*, a "task" that can be performed by any actor in the ongoing interaction or by someone or something in the environment and that has a peripheral effect on the execution of the task. The resulting model coverage can be seen in figure 2(c).

ArtStudio is a tool for model-based design, developed by Thevenin for his PhD [26]. It supports the graphical specification of a task model (based on CTT), an abstract presentation model (notation using interaction-task notation of CTT and rectangles to symbolise groupings, interactor specification is done in separate model) and domain model (using UML). All models are also described in an XML format (see also figure 2(d)). The specification of a separate interaction model feels unnatural. The abstract user interface description consequentially feels very limited; all interactors use the same representation with their name under it and are grouped using rectangular boxes.

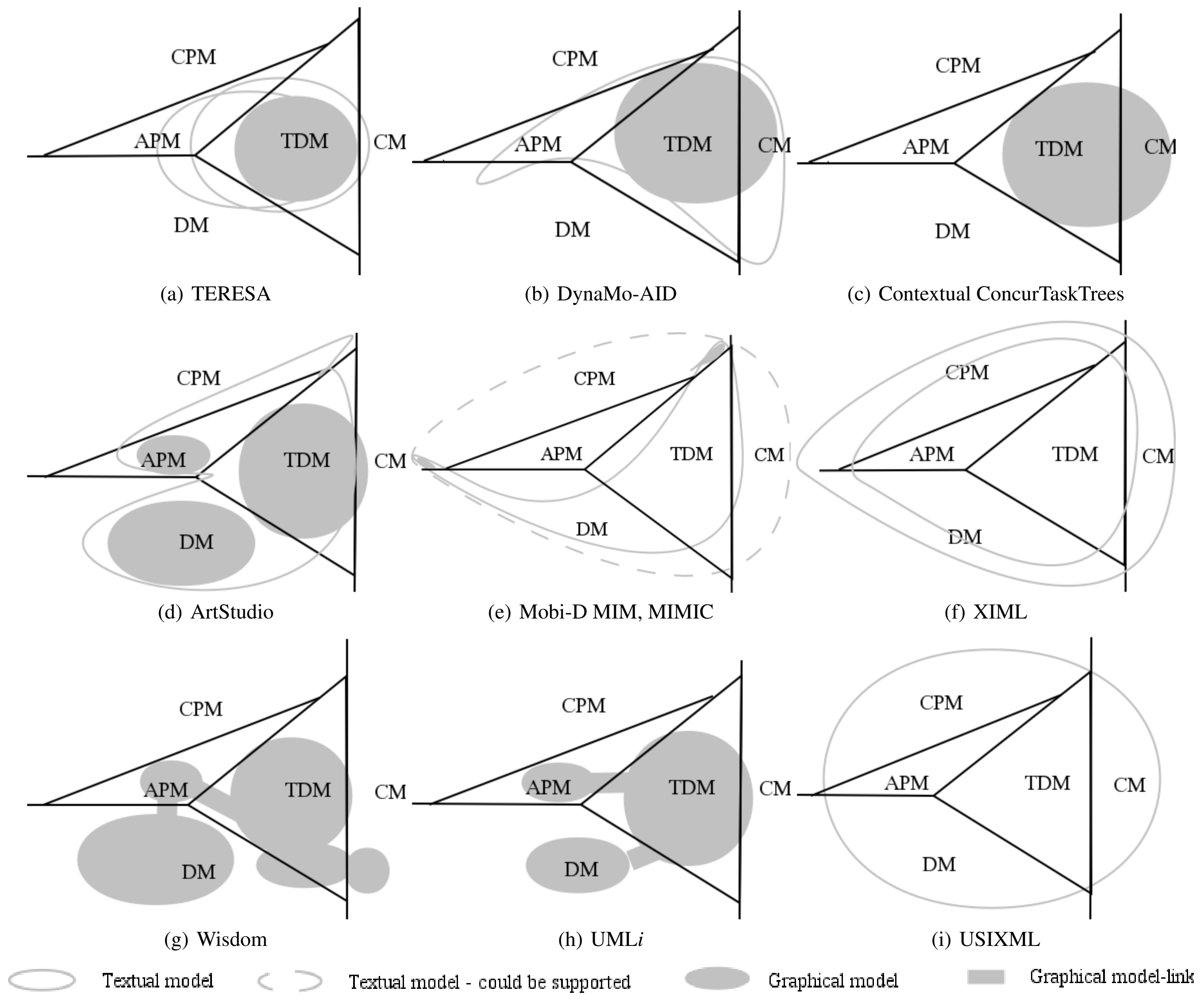

Figure 2. Models used in various model-based design approaches

Notation	User	Platform	Environment
TERESA (T: AUI, G)			
TERESA (T: CTT)		X	
DynaMo-AID (G)	Y	Y	Y
DynaMo-AID (T)	X	X	X
Contextual CTT	Y	Y	Y
ArtStudio		X	
MIM	X		
Wisdom	X		
UML*i*			
Ximl/MIMIC	X	x	x
USIXML	X	X	X
G: graphical, T: textual X: supported, Y: indicated, x: could be supported			

Table 1. Supported parts of context

Mobi-D [24] supports model-based design of user interfaces and uses five models to accomplish this: user, task and domain model as abstract models and dialogue and presentation model as concrete models. Mobi-D does not support automatic conversion between the models, but rather supports the user in making the conversion. All models and mappings are specified with a single textual notation, Mecano Interface Model (MIM), using MIMIC [22]. The tools of Mobi-D do not have a graphical representation of the supported models, although suggestions for the concrete presentation model are made graphically.

The model coverage of MIM is shown in figure 2(e), the total coverage of MIMIC is similar to that of XIML since both languages are generic modeling languages whose expressiveness can partly be defined by the user of the language. The total possible coverage of both MIMIC and XIML is shown as a dashed outline in respectively figure 2(e) and 2(f); a designer might make instantiations of all models and all relations between the models. The predefined models and relations are covered by the full gray outline. XIML [23] can be considered to be the XML-based successor of MIMIC; it has similar goals and a similar set of predefined models. Instead of full gray areas, as in the other figures, an hatched gray area is used to denote that part of the flexibility comes from possible extentions by the user of the notation; both MIMIC and XIML are meta languages. The platform model and environment model are marked as "could be supported" in table 1 because they are not supported by the models already defined in XIML, but could probably implemented using the fact that XIML is a meta-language.

Some of the approaches that have an abstract user interface model and also have a graphical representation are based on UML. The Wisdom-approach [17] is a model-based design methodology and accompagning notation for software design in small software compagnies or small groups in greater enterprises. The notation is a UML profile, meaning that it can be expressed completely by UML, but provides several extra stereotypes [1] with a corresponding notation. For the abstract user interface description, two models are used: the dialogue model expressed by a UMLified version of the ConcurTaskTrees notation and the presentation model that visualizes the user interface structure possibly containing five different basic elements — *input element* and *input collection*, *output element* and *output collection*, and *action* — organised a hierarchy of containers. The covered reference models can be seen in figure 2(g). In contrast with the figures for TERESA XML and the graphical CTT notation, this figure contains several grey ellipses indicating different diagrams used by the Wisdom notation, and some grey rectangles indicating links between diagrams that can be visualized.

Although the notation used by the Wisdom approach has its merits, it suffers from the fact that the UMLified notation of the ConcurTaskTrees is not as readable as the original [19]. Some other diagrams also do not feel natural due to the limitations of UML 1.3 [13], most notably the inability to specify structured classes, used for creating this presentation. We can also see that several models cover information in the same model in our representation, this due to the use of different models in different stages in the Wisdom design methodology. An extra tool, CanonScetch [3], enables the design of the abstract presentations using a graphical presentation, called Canonical Abstract Prototypes, that is closer what developers are used to when making a user interface design using GUI builders as Qt Designer [27].

UML*i* [9] is an extension of UML that introduces extra concepts to UML in order to make the standard *activity diagram* suitable for representing the task model (or dialogue model) and introduces a new diagram, the user interface diagram, to represent the presentation model, resulting in the model coverage in figure 2(h). UML*i* supports *inputters*, *displayers*, *editors* and *actionInvokers* as basic abstract interaction components. These components can be organized in *containers* and ultimately belong to a *free container*, which corresponds to a window or a dialogue in a graphical user interface. The extended activity diagram, which was presented as a task model, has been criticised [19] for being too low level for this purpose, but is nevertheless very comprehensible and perhaps better suited as a dialogue model. The extended activity diagram also includes explicit links with the domain and abstract presentation model.

A drawback of UML*i* is that it extends the UML Metamodel and thus can only be supported by a tool especially designed for the notation. ARGO*i* [7] is such a tool that is publically available. This in contrast to the Wisdom method that only uses standard UML mechanisms and extends the notation using stereotypes ensuring a broad tool support. UML*i* is the only model-based approach that is discussed here, offering no explicit provisions for context modeling (although e.g. user modeling has be done using UML [17]). It is mentioned here, because it is the only approach that gives special care to the definition of links between the the task/dialogue model and the domain and abstract presentation model.

[1]Stereotypes are used in UML to describe additional constraints to existing UML classes for specific purposes and are the primary way to extend the UML notation.

USIXML [1, 15] is an XML-based specification for the description of user interfaces in the widest sense of the term. In contrast to XIML and MIMIC, it chooses to define an exact framework instead of a more generic approach. It is focussed on business applications and tries to be as complete as possible in the definition of all the models that are considered to be relevant. The language allows the specification of domain, task, abstract user interface, concrete user interface and context models. Mappings and transformations between the different models can be explicitly defined in separate models: mapping, transformation and rule-term model. It is one of the most complete and in-depth specifications available as can be seen in figure 2(i).

The great amount of tools available for USIXML as seen in table 2 should be put in perspective: not all tools support USIXML completely and most are special purpose tools built especially for USIXML at UCL (Université Catholique de Louvain) except TERESA that only supports the concrete presentation.

CONCLUSIONS AND FUTURE WORK

Multiple approaches towards model-based design of user interfaces are found in litterature. In this paper, we presented requirements for a notation that supports model-based design of context sensitive user interfaces. These requirements consisted of the models that should be supported and some qualitative properties of the notations for such models. We then discussed several notations keeping these requirements in mind.

Based on the discussion we can conclude that while most notations are understandable and have tool support for the models they support, much work is needed before we can claim that model-based design can greatly ease the design of context-sensitive applications:

- While there seems a great level of agreement over the basics of a notation for the task model, the ConcurTaskTrees notation is used (and extended) in several approaches [5, 11, 28]. There is still some work to see whether and how to integrate context information in the task notation. Should this be left for tool support and the underlying notation or made explicit in the graphical representation? Should the presentation of context information be included in the task model and how?
- Most approaches do not have a context model, although they support context the definition of context information to some degree and those that cover almost all sub-models of the context model are very constrained in the amount of context they support. A more powerful context model is needed, possibly based on the work of Henricksen and Indulska [14], on the Context Ontology Language [25] or on emerging context ontologies, such as in [20].
- Support for graphical notations is still limited. Many approaches only have graphical notations for only one or two models and the models that have graphical notations for their models have received valid critics on some of their notations. Some more work is needed to ensure expressive and understandable graphical notations for all the needed models including the necessary links. The use of expressive and understandable graphical notations can be an important for a designer to get a clear understanding and overview of a model.

ACKNOWLEDGEMENTS

Our research is partly funded by the Flemish government and European Fund for Regional Development. The CoDAMoS (Context-Driven Adaptation of Mobile Services) project IWT 030320 is directly funded by the Flemish Institute for the Promotion of the Scientific-Technological Research in the Industry (IWT – Vlaanderen).

REFERENCES

1. *UsiXML*. `http://www.usixml.org`.
2. Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Nathalie Souchon, Laurent Bouillon, and Jean Vanderdonckt. Plasticity of user interfaces: A revised reference framework. In *Proceeding of TaMoDia 2002*, pages 127–134, 2002.
3. Pedro F. Campos and Nuno J. Nunes. Canonscetch: a user-centered tool for canonical abstract prototyping. In *Pre-proceedings of EHCI/DSV-IS'2004*, pages 108–126, July 11–13 2004.
4. Tim Clerckx, Kris Luyten, and Karin Coninx. Dynamo-aid: a design process and a runtime architecture for dynamic model-based user interface development. In *Pre-Proceedings of EHCI / DSV-IS'2004*, pages 142–160, July 11–13 2004.
5. Tim Clerckx, Kris Luyten, and Karin Coninx. Generating context-sensitive multiple device interfaces from design. In *Pre-proceedings of CADUI 2004*, pages 288–301, 2004.
6. Joëlle Coutaz and Gaëtan Rey. Foundations for a Theory of Contextors. In Christophe Kolski and Jean Vanderdonckt, editors, *CADUI 2002*, volume 3, pages 13–33. Kluwer Academic, 2002. Invited talk.
7. Paulo Pinheiro da Silva. *ARGOi Modelling Tool*. `http://www.cs.man.ac.uk/img/umli/software.html`.
8. Paulo Pinheiro da Silva. User interface declarative models and development environments: a survey. In *Proceedings of DSVIS 2000*, pages 207–226, 2000.
9. Paulo Pinheiro da Silva and Norman W. Paton. Object modelling of user interfaces in umli. *IEEE Software*, 20(4):62–69, July–August.
10. Anind K. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, College of Computing, Georgia Institute of Technology, December 2000.
11. Anke Dittmar and Peter Forbrig. The influence of improved task models on dialogues. In *Pre-Proceedings of CADUI'2004*, pages 1–14, 2004.

Notation	**understandable**		**split**	**not isolated**	**expressive**	**tools**
	base	spec				
CTT graphical (G)	hierarchical, LOTOS operators	Pu	only task model	links through tool	unclear which task for which platform	CTTE (1)) TERESA (2)
TERESA CTT (T)	XML	Pu	all information is organised around tasks	limited specification of other models	most information in model, context is platform	CTTE (1) TERESA (2)
TERESA AUI (T)	XML	Pu	presentation and dialogue model integrated	no relations with other models	intra-dialogue events cannot be specified	TERESA (2)
CTT, decision node (G)	hierarchical, CTT	Pu*	only task model	links through tool	decision nodes mark context dependent tasks	DynaMo-AID
CCTT (G)	context task unintuitive hierarchical	Pu*	only task model	links should be through tool	explicit notion which context is involved	no
ArtStudio (G)	TM: tree based DM: UML	Pu	separate specification of models	in underlying description?	no interactor specification in abstract UI	ArtStudio
MIM (T)	uses MIMIC	/	separate sections for models	links through design model	complex applications realised	Mobile TIMM, U-Tel model editors
MIMIC (T)	C++ syntax	Pu	not applicable	not applicable	tested with MIM	see MIM
XIML (T)	XML	Li	?	?	>= MIMIC	N/A
UML*i* (G)	UML	Pu		TDM -> DM TDM -> AM	rather low-level	ARGO*i* (3)
Wisdom (G)	UML, UML-ified CTT	Pu	different diagrams for diff. models	links between diagrams		UML tools CanonScetch
USIXML (T)	XML, CTT (T), UAPROF vocabulary	Pu	separate specification of models	links through dedicated models	limited for context specification	GraphiXML (4) VisiXML FlashiXML Tcl-Tk UsiXML RecursiXML TERESA (2) TransformiXML
G: graphical notation, T: textual notation (1) available from: `http://giove.cnuce.cnr.it/ctte.html` (2) available from: `http://giove.cnuce.cnr.it/teresa.html` (3) available from: `http://www.cs.man.ac.uk/img/umli/software.html` (4) available from: `http://www.usixml.org/index.php?page=grafixml.xml` Pu: publically available, Pu*: underlying textual notation not available, Li: restricted available						

Table 2. Qualitative evaluation of existing notations

12. Object Management Group. *UML Resource Page*. `http://www.uml.org/`.

13. Object Management Group. *OMG Unified Modeling Language Specification*. June 1999. Version 1.3.

14. Karen Henricksen and Jadwiga Indulska. A software engineering framework for context-aware pervasive computing. In *Proceedings of PERCOM 2004*, pages 77–86, 2004.

15. Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, Murielle Florins, and Daniela Trevisan. Usixml: A user interface description language for context-sensitive user interfaces. In *ACM AVI 2004 Workshop: UIXML 2004*, pages 55–62, 2004.

16. Giullio Mori, Fabio Paternò, and Carmen Santoro. Tool Support for Designing Nomadic Applications. In *Intelligent User Interfaces*, pages 141–148, January 12–15 2003.

17. Nuno Jardim Nunes and Joao Falcao e Cunha. Towards a uml profile for interaction design: the wisdom approach. In *UML 2000*, pages 117–132. Springer, 2000.

18. Fabio Paternò. *Model-Based Design and Evaluation of Interactive Applications*. Springer, 2000.

19. Fabio Paternò. Towards a uml for interactive systems. In *Proceedings of the EHCI 2001*, pages 7–18, 2001.

20. Davy Preuveneers, Jan Van den Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, Yolande Berbers, Karin Coninx, Viviane Jonckers, and Koen De Bosschere. Towards an extensible context ontology for ambient intelligence. In *Proceedings of EUSAI 2004*, 2004. Accepted for publication.

21. Costin Pribeanu, Quentin Limbourg, and Jean Vanderdonckt. Task Modelling for Context-Sensitive User Interfaces. In Chris Johnson, editor, *Interactive Systems: Design, Specification, and Verification*, volume 2220 of *LNCS*, pages 60–76. Springer, 2001.

22. Angel Puerta. The mecano project: Comprehensive and integrated support for model-based interface development. In *Computer-Aided Design of User Interfaces*, pages 19–25. Presses Universitaires de Namur, June 5–7 1996.

23. Angel Puerta and Jacob Eisenstein. Ximl: A common representation for interaction data. In *Proceedings of the IUI 2004*, pages 214–215. ACM Press, 2002.

24. Angel Puerta and David Maulsby. Management of design knowledge with mobi-d. In *Proceedings of IUI'97*, pages 249–252, January 6–9 1997.

25. Thomas Strang, Claudia Linnhoff-Popien, and Korbinian Frank. CoOL: A Context Ontology Language to enable Contextual Interoperability. In *Proceedings of DAIS2003*, pages 236–247, 2003.

26. David Thevenin. *Adaptation en Interaction Homme-Machine: le Cas de Plasticité*. PhD thesis, Université Joseph Fourier, UFR en Informatique et Mathématique Appliquées, December 21 2001.

27. Trolltech. *Qt Designer*. `http://www.trolltech.com/products/qt/designer.html`.

28. Jan Van den Bergh and Karin Coninx. Contextual concurtasktrees: Integrating dynamic contexts in task based design. In *Proceedings of PERCOM workshop CoMoRea 2004*, pages 13–17. IEEE Press, 2004.