Bridging the Gap: Empowering Use Cases with Task Models

Daniel Sinnig
Computer Science Institute
University of Rostock, Germany
dasin@informatik.uni-rostock.de

Rabeb Mizouni
Information Technology College
UAE University, Al-Ain, UAE.
mizouni@uaeu.ac.ae

Ferhat Khendek
ENCS Faculty
Concordia University, Canada.
khendek@encs.concordia.ca

ABSTRACT
Use cases have become the standard for modeling functional requirements, whereas task models are used to capture UI requirements. Despite recent advances, software engineering (SE) and user interface (UI) design methods are poorly integrated making it difficult for SE and UI teams to collaborate, synchronize their efforts and avoid inconsistencies. To address these issues, we propose an integrated development methodology for use cases and task models. Both artifacts are used to specify software requirements, but emphasize two different aspects in a complementary manner. The integration consists of using CTT task models to iteratively enrich UI related steps in the use case model. We demonstrate that such an approach allows for a clear separation of concerns and therefore avoids potential inconsistencies between the two artifacts.

Author Keywords
User interface development, use cases, task models, development methodology

ACM Classification Keywords
D.2.1 Requirements/Specifications, D.2.9 Management

General Terms
Design, Human Factors, Management

INTRODUCTION
The development of interactive systems is a multidisciplinary process requiring collaboration between several teams with different backgrounds; each bringing in its own experience and view of the system under development. **Efficient communication** and **collaboration** between software engineers and UI designers is required to develop a software product that satisfies its functional requirements *and* that is highly usable [11].

In current practice UI design methods are poorly integrated with standard software development processes [1]. Instead of having an integrated process, where UI design follows as a logical progression from a functional requirements specification, these activities are either (1) **handled independently** and therefore potentially lead to **inconsistencies** and redundancies or (2) UI and SE **artifacts** are **intermingled** into one specification **violating** the principle of **separation of concerns** [23].

The scientific literature [2] distinguishes two main strategies to close this conceptual gap: (i) achieving integration at the artifact level and (ii) achieving integration at the methodological level. The former approach is concerned with proposing unified SE artifacts and UI artifacts. The latter acknowledges the utility of heterogeneous specifications and instead, suggests integration at the process level. An effective integration, however, is not a matter of re-representing SE or UI artifacts. As Paternò points out, specialized notations should be used in a complementary manner and integrated by a common process which effectively supports software engineers and UI designers in their work rather than complicate it [17].

For example, a common mistake is to intermingle functional requirements with UI details [5]. As a result, the functional requirements specification not only becomes unnecessary long and hard to maintain, but also decisions about UI details are made too early in the process, potentially excluding more favorable alternatives. Instead, functional and UI requirements should be captured in specialized artifacts interrelated through well-defined traceability links.

This paper introduces an integrated development

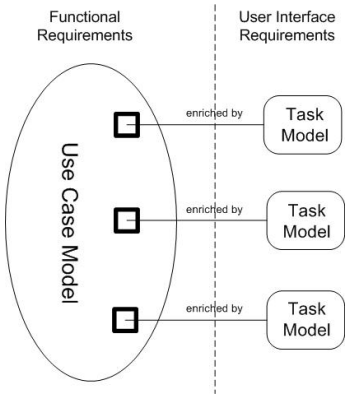

Figure 1: Empowering Use Cases with Task Models

EICS'10, June 19–23, 2010, Berlin, Germany.

Use Case: Process Contact Request

Goal: Primary actor successfully processes a contact request
Level: User-goal
Primary Actor: Customer
Main scenario
1. Customer authenticates herself/himself.
2. System ensures the validity of the information.
3. System informs customer that login was successful.
4. Customer identifies and confirms the contact request.
5. System processes the user request.
6. System informs user that a new contact was added to contact list.

Extensions
3a. The provided information cannot be validated:
3a1. The system informs the customer that the provided information could not be validated.
3a2. *Use case resumes at step 1.*
4a. Customer rejects contact request:
4a1. Customer rejects the contact request.
4a2. System informs user that the contact was not added to contact list.

Figure 2. "Process Contact Request" Use Case

methodology for use case models and task models. The former is the predominant SE artifact for specifying functional requirements and the latter is commonly used to capture UI requirements and design information. We propose to use both artifacts in a complementary manner. The core functional requirements are captured in the use case model, which is enriched by a set of task models capturing UI specific interactions where it is necessary. As depicted in Figure 1, traceability between the various models is established through 'anchors' in the use case model pointing to the respective task model counterpart.

The remainder of this paper is structured as follows: in the next section we provide the necessary background information by reiterating key characteristics of use case and task models. Next we define our proposed development methodology, followed by a review of existing integration attempts for use cases and task models. Finally, we conclude.

USE CASES AND TASK MODELING: BACKGROUND

Use Case Models

Use cases were introduced in the early 90s by Jacobson [8]. He defines a use case as a "specific way of using the system by using some part of its functionality." Use case modeling has become mainstream SE practice as a key activity in the software development process (e.g., Rational Unified Process (RUP)). There is accumulating evidence of significant benefits to customers and developers [12].

A use case describes the way a system is used by its actors to achieve their goals. Actors represent users or entities (e.g., secondary systems) that interact with the system. The primary actor, typically a user, initiates the use case in order to accomplish a pre-set goal. Secondary actors play the role of supporting the execution of the use case and may participate in the interaction later.

An example of a (user-goal level) use case in "fully-dressed" format [5] is shown in Figure 2. The use case depicts the interactions involved in processing a contact request, as is typical in social networks such as LinkedIn and Facebook. The main success scenario describes the situation in which the primary actor directly accomplishes his goal of confirming the contact request. The two extensions specify alternative scenarios, which occur when the primary actor fails to authenticate himself or refuses the contact request.

In use-case driven software development processes, including RUP, ICONIX [20] and URDAD [24], the use case model drives the development starting from the initial gathering of requirements to design, implementation, deployment and testing. The use case model is developed in an incremental manner, where an initial coarse grained model is successively revised and refined throughout the software development life cycle.

Task Models

Task modeling is a well understood technique supporting user-centered UI design [16]. It has become a central concept of many model-based UI development approaches, such as MOBI-D [19] and TERESA [18].

Task models are used at different stages of development (analysis, requirements and design). The *analysis level task model* captures the current work situation and highlights elementary domain processes as well as exposes bottlenecks and weaknesses of the problem domain. When used at the *requirements level*, task models specify the envisaged way tasks are performed using the UI under development. These tasks are further refined by *design level task models*, which for example tailor the task set to a particular target device by taking into account its interaction capabilities. Their primary purpose is to systematically capture the way users achieve a goal when interacting with the system [25]. Note that this paper focuses on requirements- and design level task models and their relationship to use cases. Analysis-level task models are out of the scope.

Several task model notations exist in the literature. ConcurTaskTrees (CTT) [16], GOMS [4], TaO Spec [7], WebTaskModel (WTM) [3], and HTA [1] are among the most popular ones. In this paper, we consider CTT. In this notation, tasks are organized hierarchically, where more complex tasks are successively decomposed into simpler sub-tasks. CTT includes a set of binary and unary temporal operators. The former are used to temporally link sibling tasks, at the same level of decomposition, whereas the latter are used to identify optional and iterative tasks. An example of CTT task specification is depicted in Figure 3. It shows the break down of a root task "Authenticate" into the sub-tasks "Provide Login Data" and "Submit".

Use Cases vs. Task Models

Use case and task models are both scenario-based and as such capture sets of usage scenarios of the system under consideration. On the one hand, a use case describes

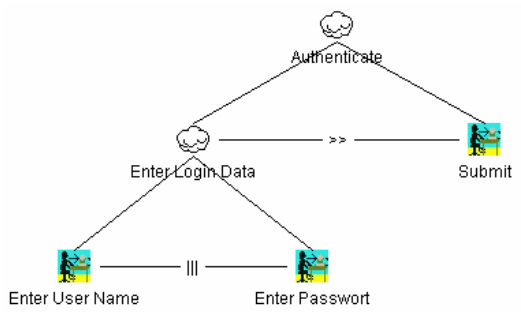

Figure 3: "Authentication" Task Model

system functionality by means of a main success scenario and extensions. On the other hand, a task model captures user-system interactions within a hierarchical task tree. We identify the following main differences that are important for understanding their intended applications and guide any integration attempt:

- In use case models, requirements are captured at a higher level of abstraction whereas task models are more detailed. The atomic actions of a task model are often lower-level UI details that are irrelevant (actually contraindicated [5]) in a use case.
- Task models focus on aspects that are relevant for UI design and as such, usage scenarios are strictly depicted as input-output relations between the user and the system. System interactions that are hidden from the end user, e.g., the involvement of secondary actors or internal computations as specified in use case models, are *not* captured.

EMPOWERING USE CASES WITH TASK MODELS

An integration of use cases and task models is not a simple matter of improving expressiveness or of the ability to convert or embed a model into another one. Instead, use case and task models should be used according to their intended purposes and integrated at the process level. This enables software engineers and UI experts to use familiar notations that support them to effectively carry out their work.

Assumptions

Before introducing our methodology we start by outlining a few necessary assumptions:

1) The functional requirements captured in the use case model are independent of a particular user interface. The "fully dressed" use case format [5] as portrayed in Figure 2 is used for describing use cases.
2) The requirements and design information captured in task models extend the functional requirements by taking into account the specifics of a particular type of user interface. For our approach we recommend using CTT task models as these provide the richest operator set and are supported by the CTTE tool [13].
3) We assume that the development work of specifying functional and UI requirements is accomplished by two distinct teams: SE team and HCI team. The efforts of both teams need to be synchronized. Unlike the approach by Lu [10] where task modeling precedes the development of use cases, we assume that task models are used at the requirements / design stage and as such are developed based on a given use case model.

Development Methodology

As shown in Figure 4 our methodology consists of three phases. (1) The SE team prepares a coarse-grained use case model, rich enough to distinguish between UI-related steps and system-related steps. UI-related steps are identified with so-called anchors points. (2) Once, the anchors have been identified, the UI team is responsible for defining the corresponding CTT task models, while the SE team further refines the use case model. This phase may require several iterations between the two teams. (3) Finally, the efforts of both teams are merged into a consolidated specification consisting of functional and UI requirements.

The three phases are described in detail hereafter. As a running example we use the "Process Contact Request" use case given in Figure 2.

Specification of a Coarse Grained Use Case Model

The goal of this phase is to obtain a conceptual understanding of the system. Key behaviors of the system are defined by a set of coarse grained use cases, the primary actors are identified, and an initial mapping to high-level business goals is performed. For each use case, the SE team provides an initial behavioral description and identifies its scope and priority. Without delving into details, the initial behavioral description documents the primary interactions that actors will perform with the system. Error cases and exceptions are only marginally captured and will be detailed further in the second phase.

For each use case, the SE team identifies a set of use case steps that require further elaboration with UI details. For this purpose, so called "anchor points" are used, whose aim is twofold: (1) To unambiguously identify UI-related use case steps and (2) to cross-reference associated CTT task models, which will be developed by the UI team in the next phase. For example, in the "Process Contact Request" use case, step 1 ("Authentication") and step 4 ("Identification of Contact Request") are marked with anchor points as shown in Figure 5. Both steps do *not* detail *how* the step-goals are achieved. These are UI-specific details and will be captured in the corresponding CTTs.

Use Case Refinement and CTT Task Modeling

For each specified anchor, the UI team defines a set of UI-related interactions in the form of CTT task models. This phase requires knowledge about the associated use case and its goal as well as a deep understanding of the nature of the application and its intended usage. The former can be obtained by tracing back to the corresponding anchor point whereas the latter requires consultation of stakeholders and related requirements artifacts. Only when both factors are

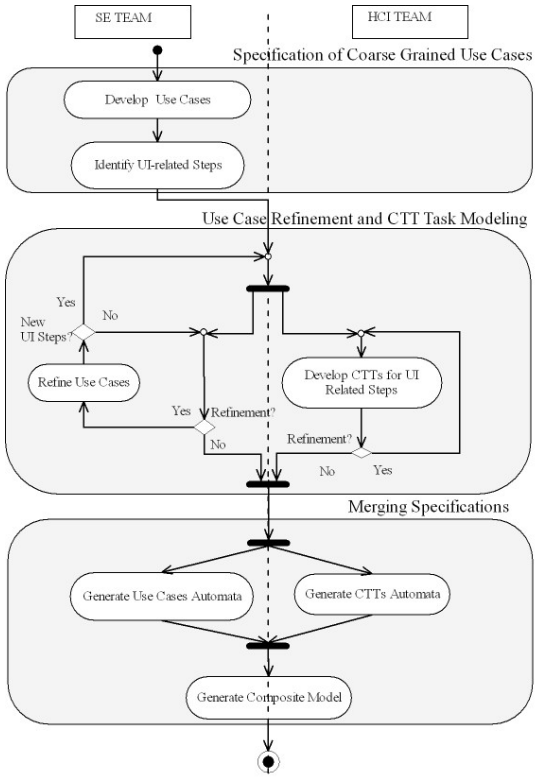

Figure 4. Collaborative Development of Use Cases and Task Models

well understood, the UI team will be able to properly evaluate the feasibility of a certain set of interactions and its usability in the context of the application. Let us recall step 1 ("Authentication") of our example use case. Depending on the UI type, the UI team may recommend any of the following interaction scenarios:

- "The user enters the user name and password in any order" (Desktop UI)
- "The user enters the user name and then the password, in this specific order." (Mobile UI)
- "The user provides her/his fingerprint." (Kiosk UI)
- "The user dictates the login information." (Text-Free Voice UI)

Once the interaction scenario is selected, the HCI team proceeds and develops a corresponding CTT task model. The various task models are then associated with the anchors in the use case, building a link between the use case steps and their UI-specific realization, as shown in Figure 5. In our example, we develop CTT task models for two anchors: use case step 1 and step 4. We assume that both Desktop UI and Text-Free Voice UI interactions are possible scenarios for the authentication step. The CTTs of these two alternatives are composed using the CTT choice operator ([]), as illustrated by "CuAu-CTT" in Figure 5. Additionally, we associate step 4 with a refining CTT labeled "ConfReq-CTT", specifying the appropriate interaction for confirming a contact request.

Concurrently to the work of the HCI team, the SE team proceeds with the elaboration of the coarse-grained use cases defined in the first phase. Thus far alternative and failure cases have been considered only marginally. However, in order to obtain a complete behavioral description, exhaustive modeling of alternative scenarios is necessary. Therefore, in this phase, the use cases are further completed by specifying the various use case extensions. Furthermore, non-UI related steps may be refined by interactions with secondary actors and/or internal system steps. Notice that both step types are irrelevant for UI design and as such should never be attributed an anchor point.

The refinement phase is iterative and requires synchronization among the involved SE and HCI teams. For example, a newly discovered use case extension (alternative scenario) may include use case steps which are UI-related and as such require enrichment with CTT task models. Similarly, certain UI-related interactions may require the availability of a system functionality, which has not yet been captured in the use case model.

Merging Specifications

In order to obtain a composite model the interactions entailed in the use cases and CTTs are merged. Intuitively, the behavior of the composite model can be summarized as follows: At first, the composite model adopts the behavior of the use case model up until a point where a step with an anchor point is encountered (e.g., step 1 in the example). At this point, the composite model adopts the behavior of the associated CTT model depicting how the primary actor may accomplish the step-goal using a particular UI. Thereafter, the composite model again, continues with the behavior of the use case model. This alternating continues until the usage scenario comes to an end.

Discussion

Our development methodology enforces clear separation of concerns and thus minimizes inconsistencies and overlaps, and strengthens the synergy between the two artifacts. At the same time (and as convenient side-effects) our methodology addresses current problems with task models and use cases, as briefly discussed next.

Task model scalability: In current practice, task models capture the behavioral aspects of the UI within a single monolithic task tree [22]. A monolithic task tree is suitable for small applications but becomes unmanageable (in terms of visualization, comprehension and modification) for applications of even moderate size. In order to reduce the complexity of task models, a modular approach is needed where instead of a single task tree a set of task trees is defined. As it turns out, our methodology promotes precisely such a modular setup. Instead of a single task model, a set of task models is defined and related to the various UI-specific use case steps that represent their context.

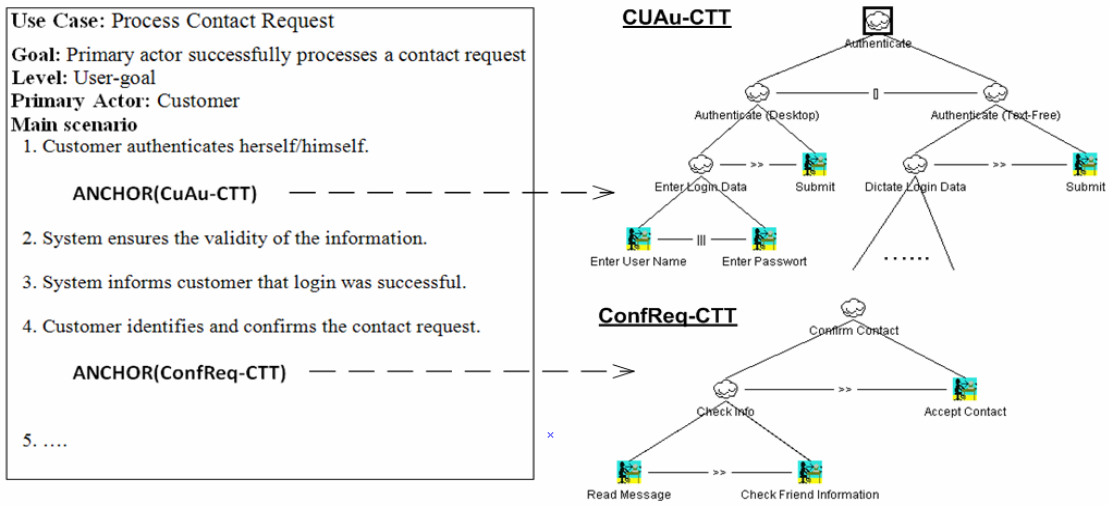

Figure 5: "Process Contact Request" Use Case Empowered with CTT Task Models

Use Case Misuse: As stated previously, use cases are frequently misemployed by intermingling the functional requirements with UI details. Our methodology acknowledges this close relationship between system functionality and the UI, but promotes capturing both aspects in separate artifacts interrelated through well-defined traceability links.

RELATED WORK

The relationship between use cases and task models has been investigated by several research groups. The integration of both artifacts is seen as a promising solution to close the gap between SE and HCI. According to Artim [2], the majority of integration attempts falls into one of the following three categories: (1) *Conversions* from task models to use case models and vice versa, (2) *Extensions* of existing SE/HCI models to capture task model and use case information respectively, and (3) *Methodological approaches* which attempt the integration at the process level. In what follows we review for each category the most significant attempts:

Paternò [17] proposes a method for integrating use case diagrams and task models. Use cases denote core functionalities offered by the system which are refined by a set of task models. However, the scenario descriptions entailed in each use case are not taken into account. Another approach that falls under the first category is presented by Noberga *et al*. [14]. Motivated by the fact that the current UML standard provides insufficient support for modeling interactive systems, a mapping from CTT task models to UML activity diagrams is proposed. The mapping is complemented by an extension of UML with high-level syntactic constructs related to task modeling.

Da Silva and Paton [6] propose UML*i* as a modeling language for interactive systems. UML*i* extends UML with UI diagrams for describing abstract interaction objects. According to their eight-step methodology, use cases are employed to define high-level functionalities which are further refined by a set of user tasks captured in extended UML activity diagrams. A set of logical links placed between the various use cases and the activity diagrams establish traceability between UI details and the corresponding functional requirements.

Rosson and Carroll [21] propose a scenario-based approach to object-oriented analysis and design. In order to integrate usability concerns with functional modeling, the existing or envisioned system is modeled by a set of *instance scenarios*. In a bottom-up approach the various scenarios are processed and serve as a basis for the creation of the object model. Nunes and Conha [15] point out that UML provides inadequate support for modeling architectural concerns of interactive systems and propose their Wisdom framework to fill this gap. While mainly based on existing UML models, Wisdom introduces a CTT-like notation to capture the envisioned dialogue between users and the application.

The RESCUE requirements management process [9] is an integration attempt that falls under the third category. The process consists of a number of sub-processes. At the analysis stage, human activity modeling (a form of task modeling) is carried out to form an understanding of the current socio-technical system in terms of users' tasks, goals and domain concepts. Based on the information gained, a system-goal model using the i* approach [26] and a set of use cases of the envisioned system are derived. Subsequent phases comprise the development of system architectural models and the specification of detailed interactions between users and the UI. Consistency across artifacts is ensured through a number of checks that are (manually) conducted at the end of each phase.

The work reported in this paper was conducted in the context of a larger project involving the formalization of use case models and task models [22] based on a common semantic foundation. In our original work, we proposed employing use case and task models asynchronously

specifying two independent, yet overlapping views of the system. In order to ensure that both views were consistent, a set of equivalence and refinement relations are proposed. In this work we are using both artifacts in a complementary manner enforcing strict separation of concerns, which has the advantage that consistency is an intrinsic property of the development process.

CONCLUSION

In this paper, we proposed a development methodology for combining use case and task models in a complementary fashion. Following a three-phase process, a set of coarse grained use cases, expressing the raw functional requirements, are successively enriched with modular CTT specifications resulting in a composite system model. The composite system model represents both functional and UI-specific requirements with a concise relationship.

We believe that the proposed methodology will help in narrowing the gap between SE and HCI. Synchronization of the efforts of SE and HCI teams in a complementary fashion is likely to have a positive impact on software quality and usability. Use cases and task models are strictly employed according to their intended purposes. We enforce a clear separation of concerns resulting in a requirements specification that is easier to maintain and manage. Through such integration, we also conveniently circumvent the problem of task-model scalability. Moreover, the composite system model can be used as a reference specification for integrated testing, verification and validation purposes.

The formal semantics of the composite model is being defined and is based on our previous work on common semantics for use case models and task models [22]. Furthermore, we plan to carry out comprehensive case studies and apply our approach to industrial-size projects.

REFERENCES

1. Annett, J. and Duncan, K. D., Task Analysis and Training Design, in Occupational Psychology, **41**, pp. 211-221, 1967.
2. Artim, J. M. (1997). Integrating User Interface design and Object-Oriented Development through Task Analysis and Use Cases. CHI97 - Object Models in User Interface Design Workshop. Atlanta, GA.
3. Bomsdorf, B., The WebTaskModel Approach to Web Process Modelling, in Proc. of *TaMoDia'07,* Toulouse, France, pp. 240-253, 2007.
4. Card, S., Moran, T. P. and Newell, A., The Psychology of Human Computer Interaction, 1983.
5. Cockburn, A., Writing Effective Use Cases, Addison-Wesley, Boston, 2001.
6. de Paula, M. i. G., da Silva, B. S. and Barbosa, S. D. J., Using an interaction model as a resource for communication in design, in Proc. of CHI '05, pp. 1713-1716, Portland, OR, 2005.
7. Dittmar, A., Forbrig, F., Stoiber, S. and Stary, C., Tool Support for Task Modelling - A Constructive Exploration, in Proc. of *DSV-IS*, Hamburg, Germany, pp. 59-76, 2004.
8. Jacobson, I., Object-Oriented Software Engineering: A Use Case Driven Approach, ACM Press, NY, 1992.
9. Jones, S. and Maiden, N., RESCUE: An Integrated Method for Specifying Requirements for Complex Socio-Technical Systems, chapter in *Requirements Engineering for Sociotechnical Systems*, Information Science Publishing, pp**.** 245-265, 2004.
10. Lu, S., Paris, C., Linden, K. V. and Colineau, N., Generating UML Diagrams from Task Models, in Proc. of *CHINZ'03*, Dunedin, New Zealand, pp. 9-14, 2003.
11. Memmel, T. and Reiterer, H., Model-Based and Prototyping-Driven User Interface Specification to Support Collaboration and Creativity, in Journal of Universal Computer Science, **14** (**19**), 2008.
12. Merrick, P. and Barrow, P., The Rationale for OO Associations in Use Case Modelling, in Journal of Object Technology, **4** (**9**), pp. 123-142, 2005.
13. Mori, G., Patern, F. and Santoro, C., CTTE: Support for Developing and Analyzing Task Models for Interactive System Design, in IEEE Transactions on Software Engineering, **28** (**8**), pp. 797-813, 2002.
14. Nobrega, L., Nunes, N. J. and Coelho, H., Mapping ConcurTaskTrees into UML 2.0, in Gilroy, S.W., Harrison, M.D. (eds) Interactive System, 3941, 2006.
15. Nunes, N. J. and Cunha, J. F., Wisdom ? A UML Based Architecture for Interactive Systems, in Proc. of *DSV-IS'01*, Glasgow, Scotland, UK, pp. 191-205, 2001.
16. Paternò, F., Model-Based Design and Evaluation of Interactive Applications, Springer, 2000.
17. Paternò, F., Towards a UML for Interactive Systems, in Proc. of *Engineering HCI '01*, Toronto, Canada, Springer Verlag, pp. 7-18, 2001.
18. Paternò, F., Santoro, C., Mäntyjärvi, J., Mori, G. and Sansone, S., Authoring Pervasive MultiModal User Interfaces, in International Journal of Web Engineering and Technology, pp. 235-261, 2008.
19. Puerta, A., A Model-Based Interface Development Environment in IEEE Software, **14**(**4**), pp. 40-47, 1997.
20. Rosenberg, D. and Stephens, M., Use Case Driven Object Modeling with UML: ICONIX Process in Theory and Practice, Addison-Wesley, 2006.
21. Rosson, M., Integrating development of task and object models, in Communications of the ACM, **42** (**1**), 1999.
22. Sinnig, D., Chalin, P. and Khendek, F., Common Semantics for Use Cases and Task Models, in Proc. of *IFM'07*, Oxford, England, pp. 579-598, 2007.
23. Sinnig, D., Chalin, P. and Khendek, F., Consistency between Task Models and Use Cases, in Proc. of DSV-IS'07, Spain, pp**.** 71-88, 2008.
24. Solms, F., Business Process Modeling using URDAD, Technical Report in Solms Consulting, 2005.
25. Souchon, N., Limbourg, Q., Vanderdonckt, J. and V, J., Task Modelling in Multiple Contexts of Use, in Proc. of DSV-IS'02, pp. 59-73, Rostock, Germany, 2002.
26. Yu, E., Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering, in Proc. of *Int. Symp. on Requirements Engineering*, pp. 226-235, 1997.