User-Centered CASE Tools for User-Centered Information Systems

Pedro Campos

University of Madeira
Campus da Penteada, 9000-390 Funchal, Portugal
pcampos@uma.pt

Abstract. The goal of this research is to design and evaluate a new generation of CASE tools, capable of clearly supporting all the styles of work adopted by teams of developers and capable of promoting Usability concerns to Information Systems building. Our research approach is based on (a) creating usable and effective models of the developers behavior, and (b) building innovative, unconstrained collaborative prototypes that allow co-authoring of models by all stakeholders involved (programmers, marketers and interaction designers).

1 Problem Formulation and Importance

CASE (Computer-Aided Software/Systems Engineering) tools are supposed to increase productivity, improve the software product quality and make Information Systems (IS) development a more enjoyable task [7]. However, they have been failing to deliver the benefits they promise [6]. Previous research [8] reports that one year after introduction, 70% of the CASE tools are never used, 25% are used only by a limited number of people in the organization, and 5% are widely used but not to capacity.

One of the goals of the UML was to encourage the growth of the object tool market, enabling tool interoperability at semantic level and providing a common language for specifying, visualizing and documenting software systems. In the 90s, the International Data Corporation (IDC) reported optimistic predictions for the worldwide market for modeling tools, with a compound annual growth of 54.6% from $127.4 million in 1995 to $1,125.2 million in the year 2000. Much of that growth was expected to stem from adoption by small software development organizations. However, those expectations failed completely and today the IDC expects that worldwide revenues on the tools market will stay flat, growing slowly until 2006. Among those problems is adequate support for multiple stakeholders involved in the software development process since early inception and traceability between the early requirement level models and the end-product artifacts.

An unconstrained modeling environment should be part of the next generation of CASE tools, aiming at supporting developers in solving problems and reasoning about all phases of IS development [7, 11]. In order to find a tighter fit between human experiences and IS engineering practices, the developer's behavior has to be studied and modeled [7, 11]. As an example, Damm et al. [4] describe Ideo-

gramicUML, a tool based on a direct, white-board-like interaction achieved using gesture input on a large electronic whiteboard. They argue that the conflicting advantages and disadvantages of whiteboards and modeling tools can lead to frustrating and time consuming switches between the two technologies, and therefore aim at offering a tool capable of offering the best of both worlds.

Also recognizing that software engineers work together in a variety of styles and move frequently between these styles throughout the course of their work, Wu and Graham [12] propose a tool called the Software Design Board, a collaborative design tool that supports a variety of styles of collaboration and facilitates transitions between asynchronous and synchronous styles of collaboration, and between co-located and distributed styles of collaboration. The whiteboard space can be divided into any number of segments, which allow data to be shared in different ways.

These examples demonstrate that there is a current trend towards building a new generation of user-centered CASE tools that better support the IS building activities.

2 Proposed Approach

We argue that in order to achieve a stronger market acceptance of CASE tools, a new generation of developer-centric tools will have to emerge, tools capable of clearly supporting the developer's activities and styles of work. In order to solve this problem we need both usable and expressive notations and tools that enable the creation of IS specifications that leverage the design and thought process [9].

This research has focused on creating a model that could effectively and easily capture the different styles of work adopted by software engineers, and using that model to drive the development of better User-Centered Development (UCD) tools. We propose a new workstyle model that can be used to estimate the stage/effort of development in a graphically intuitive way as well as to drive the development of new UCD and CASE tools (see Section 3). Thus, we claim that the model can be *effectively* and *easily* used to capture the IS development activities and workstyles of all stakeholders involved. We argue it can act as a tool for driving the development of a new class of CASE tools. In order to measure this, we plan to apply it to a wide range of tools in diverse categories, evaluate the model's usage by several developers and use the model to specify requirements for CASE tools.

Requirements management and elicitation is widely recognized to be one of the major problems in modern IS development. This stage of development involves multiple stakeholders, usually with different backgrounds, and is currently faced with the advent of multi-platform development. In this context, new tools are required to enable cooperation and communication of multiple stakeholders over a common semantic model that is capable of driving modern software development.

By modeling the behavior and styles of work of all stakeholders involved in IS development, we expect to achieve a better framework for driving the design of new CASE tools. We also expect to develop prototypes capable of illustrating the main advantages of the proposed new kind of tool.

Also, this project aims at integrating cross-disciplinary research from the service marketing, software engineering and usability engineering fields. There is substantial

evidence that marketing plays a major role in driving modern software development, and we expect to provide methodological and technological support that could benefit areas that are of ultimate importance for companies offering innovative services through multi-distribution channels. More precisely, we aim at exploring new ways of combining different representations used in different fields in a way that promotes co-authoring and co-evolutionary development of requirements over a common semantic model.

3 Research Results to Date

The proposed workstyle model incorporates two *collaboration*-style axes: Asynchrony and Distribution; three *notation*-style axes: Perspective, Formality and Detail; and three *tool-usage* related dimensions: Stability, Functionality and Traceability, as shown in Figure 1. Here, I briefly describe the meaning of each of these dimensions.

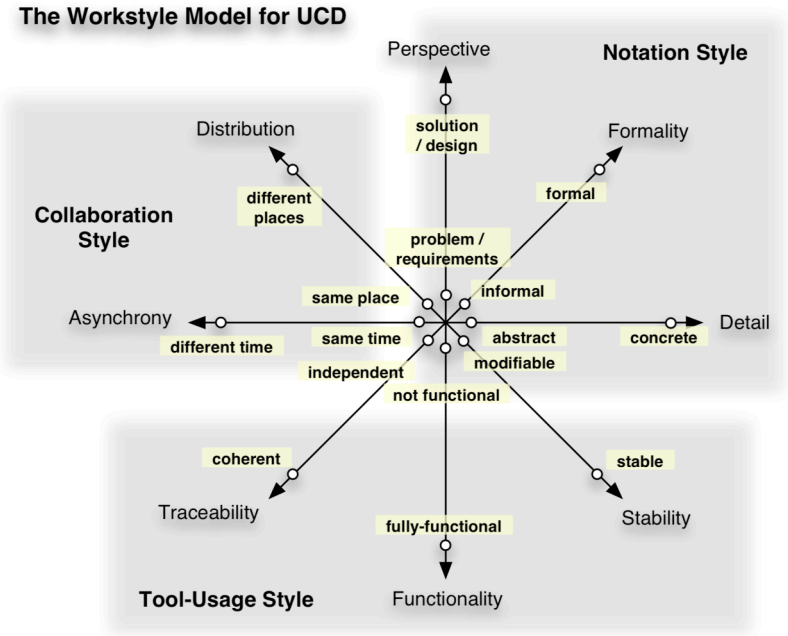

Fig. 1. A workstyle model for user-centered information systems development.

Perspective. This axis plots the perspective, or view, of the artifact being developed (from problem/requirements elicitation to solution/final design).

Formality. This axis classifies the workstyle of a developer creating artifacts in a formal vs. informal way. In the early stage of the process, developers use rough, ambiguous sketches to freely express ideas quickly [12]. As design progresses, a more formal style of work is incrementally adopted, as developers need to focus on the

precise meaning of their models. An example of this shift is moving from a whiteboard to a CASE tool [12].

Detail. This axis plots the level of detail (or abstraction) the designer is working at. High-level, abstract models facilitate problem solving in organization, navigation and overall structure of the UI, leaving aside the details. On the other hand, realistic (or figurative) prototypes address high-detail design issues [3]. Disciplined designers tend to assume a workstyle that goes from higher-level abstract representations towards more realistic and detailed representations as the process evolves [3].

Stability. This dimension describes how difficult/frequent it is to modify any aspect of the artifact(s) being developed.

Traceability. This dimension describes whether the elements of the artifact being developed are consistent and interconnected (thus being highly traceable) or if they are completely unrelated and independent. As an example, developers might adopt a workstyle in which they choose to keep links from task cases steps and the concrete UI widgets that implement those task steps. In this case, it is possible to trace a task step to the concrete widget and to trace a widget to the task step it implements (a workstyle with high traceability).

Functionality. This dimension represents how much functionality is being addressed (by using the tool to build a prototype). There is a barrier between software engineers and usability professionals regarding this matter: software engineers are engaged into building reliable, functional systems, leaving user-friendliness to the usability specialists. Usability and interaction designers, on the other hand, first design and test the interface with end-users, leaving implementation to software engineers, regarded as functionality builders. Those two processes should not be separated [11] and considering this dimension will help overcome that barrier.

Asynchrony. This axis refers to the collaboration style that designers assume: they can make changes to the work being developed at the same time (a *synchronous* workstyle) or they can work at different times (engaging in an *asynchronous* workstyle) [13]. The higher the value in this axis, the more asynchronous is the workstyle.

Distribution. This dimension describes whether work is being conducted at the same physical location or at geographically distant locations.

These dimensions can be effectively used to assess a given workstyle adopted by a designer, a marketer, a developer or a team. A single workstyle is plotted as a line (a point in the eight-dimensional space) whereas regions (or planes) represent sets of workstyles. Our model can be used to identify adequate tools for a given project phase. It also serves as a discussion tool for guiding the design of new CASE tools.

Figure 2 shows IdeogramicUML plotted in our model. It covers a considerable region, which suggests its adequacy to several styles of work during IS development. IdeogramicUML only supports the syntax of the UML. It covers part of the perspective, abstraction and modifiability axis. Traceability exists to some extent, since some views are interconnected automatically and there is something like a model navigator. More differences come in the collaboration-style dimensions. IdeogramicUML uses a sketch recognition language and can be effectively used in electronic whiteboards. There is also a distributed version with awareness mechanisms built in.

Figure 2 also shows how this model was used to drive the development of a new user-centered tool for designing UI's (User Interfaces). CanonSketch [1, 2] is an UML-based tool that supports multiple levels of detail by providing the designer with three views: UML view of the UI and domain models, Canonical Abstract Prototype [3] and HTML concrete prototype (as the right side of Figure 2 exemplifies). The first two views are synchronized and the UML semantic model is used to support traceability. There is also a collaborative version of this tool in which designers can work at the same time on the model and at different places. However, support for distribution is still limited (for instance, there are no awareness mechanisms). Therefore, CanonSketch supports a region in my model, as illustrated in Figure 2.

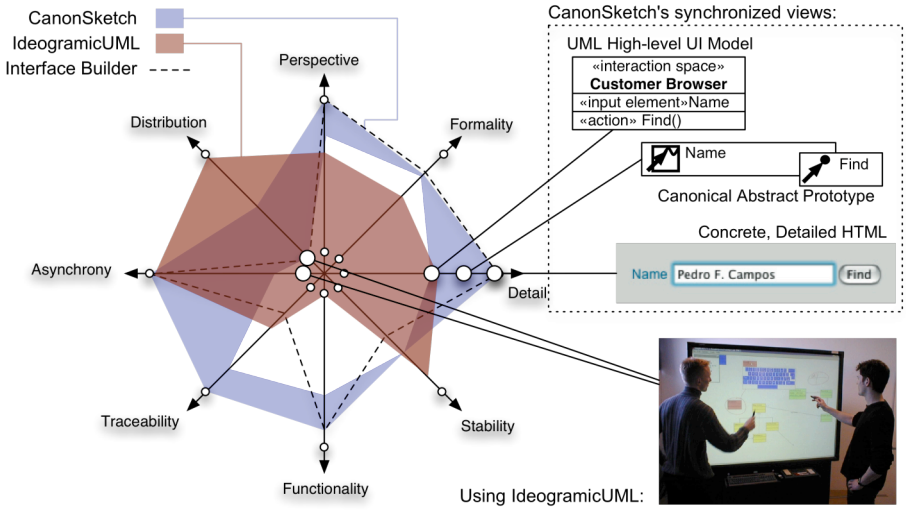

Fig. 2. CanonSketch, IdeogramicUML and a visual Interface Builder plotted in our model.

In this way, the tool seamlessly supports designers while switching from high-level abstract views of the UI and low-level concrete realizations [1]. CanonSketch has been tested under a laboratorial setting and has lead to promising results[1]. By contrast, a visual Interface Builder only supports a line in the workstyle model (the dashed line in Figure 2).

4.1 Supporting Multiple Levels of Detail in CanonSketch

CanonSketch's main language is an extension to the UML for the design of interactive systems: the Wisdom profile [10]. The Wisdom UML view is at the lowest

[1] For more information, please refer to [1, 2] or visit the tools' website, which contains publications, videos and screenshots: `http://dme2.uma.pt/canonsketch`.

level of detail: it is useful for drawing navigation and containment relationships without concern for details of layout or spatial positioning.

The HTML concrete view is at the highest level of detail: it shows a partially functional HTML prototype rendered in an embedded browser, thus allowing fast testing of the UI. The intermediate view is the Canonical Abstract Prototype view.

The symbolic notation underlying Canonical Abstract Prototypes is built from two generic, extensible universal symbols or glyphs: a generic material or container, represented by a square box and a generic tool or action, represented by an arrow. Materials represent content, information, data or other UI objects manipulated or presented to the user during the course of a task. Tools represent operators, mechanisms or controls that can be used to manipulate or transform materials [3]. By combining these two classes of components, one can generate a third class of generic components, called a hybrid or active material, which represents any component with characteristics of both composing elements, such as a text entry box (a UI element presenting information that can also be edited or entered).

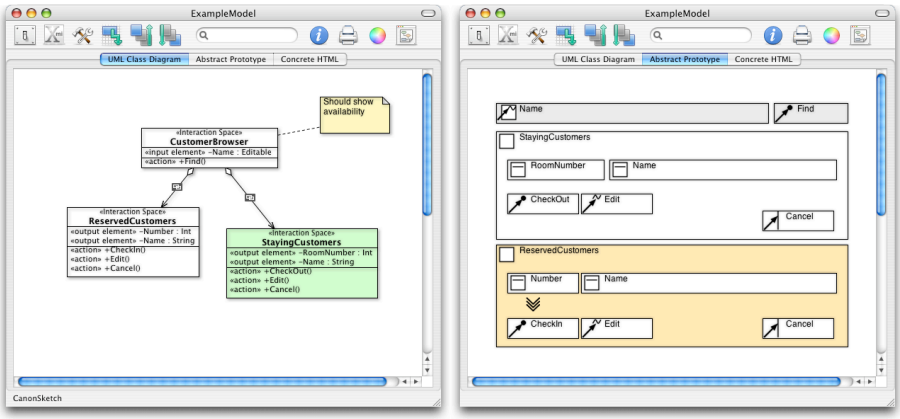

Fig. 3. CanonSketch's screenshots, showing the same model at different abstraction views: the UML view (left) and the Canonical Abstract Prototype view (right).

Figure 3 shows a screenshot of a document being edited in CanonSketch. The left part shows the UML view and the right view shows the corresponding, synchronized, Canonical Abstract Prototype. We can see, for instance, that a containment relationship between two «interaction space» classes corresponds to a Canonical material nested inside a Canonical container. The complete, precise mapping between the three views can be found in [2].

Having a single tool to manipulate the domain (or business) model as well as editing the UI by using the UML meta-model allows, for instance, achieving tool interoperability at semantic level by exporting/importing models in OMG's standard XMI format. This functionality has been partially implemented and tested.

4.2 Supporting a Task-Driven Methodology in TaskSketch

Another tool being developed and evaluated is TaskSketch. It focuses on linking and tracing use cases to the conceptual architecture of an IS. The idea is to use the Wisdom extension to the UML [10], which can be summarized as the UML class stereotypes in the lower right part of Figure 4: «Interaction Space» (models the interaction between the IS and human users within the user interface of that IS); «Task» (models the structure of the dialogue between the user and the system in terms of meaningful and complete sets of actions required to achieve a goal); «Control» (encapsulates complex derivations and calculations, such as business logic, that cannot be related to specific entity classes) and «Entity» (models perdurable, often persistent, information).

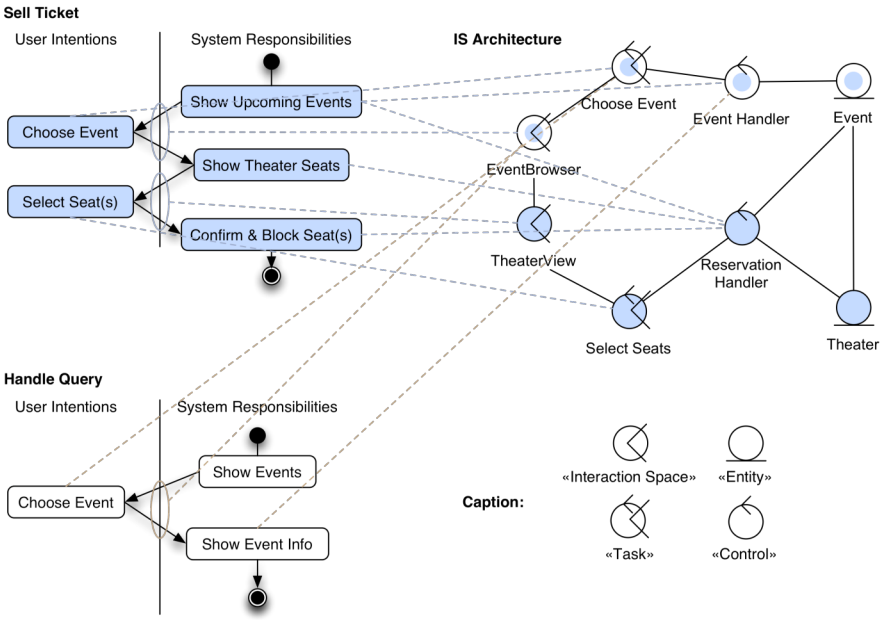

Fig. 4. Tracing use cases and task activities to the conceptual architecture of an Information System: a simple example of an Arts Center ticket selling IS.

Figure 4 shows a simple example of two UML activity diagrams representing task flows of two distinct use cases in an Arts Center ticket selling IS[2]: Sell Ticket (represented in blue) and Handle Query (represented in orange). Each use case is detailed using two swim lanes: User Intentions and System Responsibilities (the tool will also provide a participatory view and a narrative view similar to using an index card as described in [3]). For example, in the use case "Sell Ticket", it is the system responsibility to show upcoming events so that the user (the ticket selling agent) chooses one

[2] This example was provided by L. Constantine and was thoroughly used and implemented in both MSc. and BSc. HCI courses at the University of Madeira.

of them. The system then shows the available theater seats for that event and the user selects the desired seats, which are blocked by the IS. Each crossing of the swim lane originates an interaction space (Event Browser and Theater View in this example). Each action on the User Intentions swim lane corresponds to a task and each action on the System Responsibilities swim lane is associated with a control. Figure 4 describes these relations for these two distinct use cases. Using color, the developer can look at the architectural view of the system and see which classes handle which use cases. This simple support to requirements traceability can be very powerful for, e.g., prioritizing development by deciding which classes are more urgent to implement. Figure 5 shows a screenshot of the initial version of the tool.

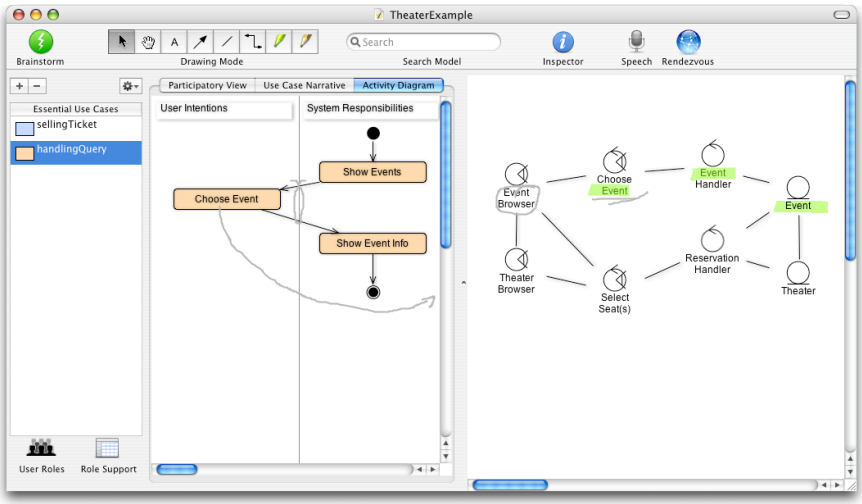

Fig. 5. Screenshot of the development environment in the initial version of TaskSketch.

We are also concentrating effort on exploring the possibilities offered by gesture recognition, mixing formal and informal notations and collaborative development using speech recognition and a shared display.

In this context, there is evidence [5] that real-time collaboration tools incorporating speech recognition and displaying information about a group's dynamics can positively impact the group's interaction. In some decision tasks, in particular during requirements elicitation and finding the core classes of an IS, there is a risk that some stakeholders holding important information will not effectively share it, thus leading the team to less informed discussions.

In the "Brainstorm Environment" we propose (as part of the TaskSketch tool mentioned above), each stakeholder is associated a color and types in ideas for core classes/concepts or requirements of the IS being developed. Every time someone sends a concept to the screen, a shape color-coded by the user who sent it starts slowly falling through the center of the window.

The content of this shared display can be manipulated by anyone, so it becomes useful to cluster concepts manually. Dragging a shape to the left or right sides of the window makes it stop falling. Concepts that remained untouched become grouped in the bottom of the window, in the same way as in [5]. Clustering of concepts can also

be made automatically, because this system uses a thesaurus and every time someone sends a common concept, such as “client” and “customer”, the two shapes become aggregated. The speech recognition system is set to dynamically recognize any of the phrases or words in the shared display. Every time a concept is recognized, the shape shows a number, which counts the number of times that concept has been spoken during the meeting. Figure 6 shows a screenshot of this environment.

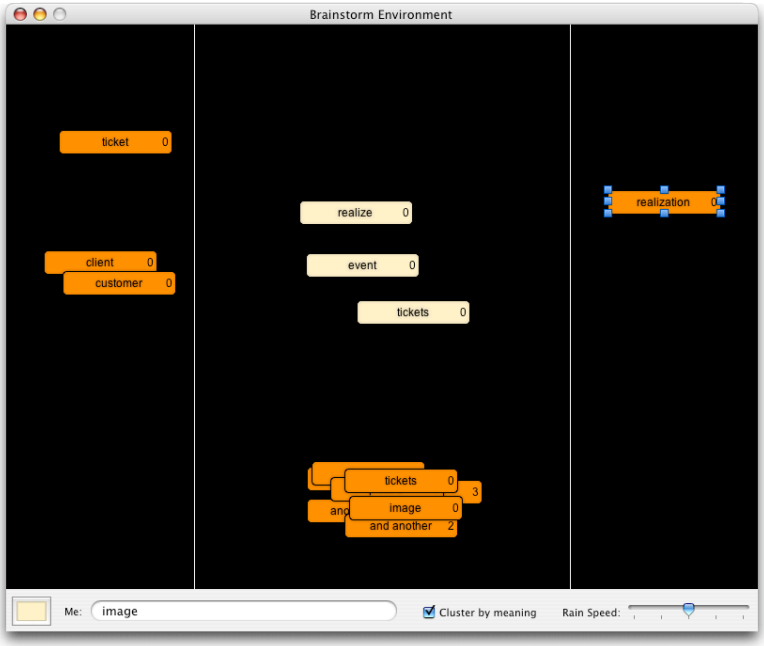

Fig. 6. Screenshot of the proposed brainstorm environment for collaborative core concepts discussion and clustering.

Users can also create “concept bins”, which act as context providers for concepts. Dragging a concept to a bin associates it with that context. Thus, it is possible to set the speech recognition system to recognize concepts only within a given context.

We foresee the following usage scenario for an environment like this one: different stakeholders meet to discuss and identify the main concepts (or classes) of the IS to develop. This includes clients, marketers, programmers and interaction designers. Each uses a microphone and has its own color. As they suggest ideas, they watch them fall and the display becomes color-filled. Thus, this system will attempt to increase the discussion of ideas as well as to foster collaboration between people with different backgrounds through an engaging environment. It is also expected that under-speakers will participate more and over-speakers will participate less, like [5] have shown. However, by the end of the meeting, it is also expected to achieve a better clustering of concepts as well as have an idea of what concepts are more important (by looking at which concepts were more referred to during the meeting).

4 Conclusion and Expected Contributions

This thesis proposal will lead to better models of the developers' behavior and to a better understanding of CASE tools usage and IS development activities. More importantly, the development of several innovative prototype tools such as CanonSketch [1, 2] has already generated interest from both industry and academia and the approach looks promising. The tools developed during this thesis work are expected to leverage the collaborative thought process, increase the quality of the developer's team decision-making and lead to more usable, user-centered Information Systems.

References

1. Campos, P. and Nunes, N. J. (2004). CanonSketch: a User-Centered Tool for Canonical Abstract Prototyping. In Proceedings of the International Conference on Engineering Human-Computer Interaction / International Workshop on Design, Specification and Verification of Interactive Systems, Springer-Verlag.
2. Campos, P. and Nunes, N. (2005). A UML-Based Tool for Designing User Interfaces. In UML Modeling Languages and Applications: UML 2004 Satellite Activities, Lisbon, Portugal, 2004, Revised Selected Papers, Springer-Verlag LNCS Vol. 3297.
3. Constantine, L. and Lockwood, L. (1999). Software for Use. A Practical Guide to the Models and Methods of Usage-Centered Design. Addison-Wesley, Reading, MA, pp. 194-195.
4. Damm, C. H., Hansen, K. M. and Thomsen, M. (2000). Tool Support for Cooperative Object-Oriented Design: Gesture Based Modeling on an Electronic Whiteboard. In Proceedings of CHI 2000, The Hague, Netherlands.
5. DiMicco, J. M., Pandolfo, A. and Bender, W. (2004). Influencing Group Participation with a Shared Display. ACM Conference on Computer Supported Cooperative Work (CSCW 2004). Chicago, IL.
6. IIvari, J. (1996). Why are CASE Tools Not Used? Communications of the ACM, 39:94-103.
7. Jarzabek, S. and Huang, R. (1998). The case for User-Centered CASE tools. Communications of the ACM, 41(8): 93-99.
8. Kemerer, C. F. (1992). How the Learning Curve Affects CASE Tool Adoption. IEEE Software, 9, 23-28.
9. Nunes, N. J. and Campos, P. (2004). Towards Usable Analysis, Design and Modeling Tools. In *Proceedings of the IUI/CADUI'04 Workshop on making model-based UI design practical: usable and open methods and tools*, Funchal, Portugal.
10. Nunes, N. J., Cunha, J. F. (2001). WISDOM: Whitewater Interactive System Development with Object Models. In Mark van Harmelen (Editor.), Object-oriented User Interface Design, Addison-Wesley, Object Technology Series.
11. Seffah, A. and Metzker, E. (2004). The Obstacles and Myths of Usability and Software Engineering. Communications of the ACM, 47(12): 71-76.
12. Wu, J. and Graham, T. C. N. (2004). The Software Design Board: a Tool supporting Workstyle Transitions in Collaborative Software Design. In Proceedings of EHCI/DSV-IS' 2004, Hamburg, Germany.