

The Knight Project: Supporting Collaboration in Object-Oriented Analysis and Design

Klaus Marius Hansen

Department of Computer Science, University of Aarhus, Åbogade 34, DK-8200 Aarhus N, Denmark
marius@daimi.au.dk



Abstract. Object-oriented analysis and design benefits from active collaboration between developers and users. Based on user studies of collaborative modelling practice, we are developing a tool, called *Knight*, for supporting this active collaboration. By combining a large shared screen with gesture recognition and zoomable interfaces, Knight allows the collaborative construction of UML based models mixed with informal drawings. This position paper thus presents a novel approach to narrowing the gap between developers and users.

1 Position

We believe that analysis, design, and implementation of software systems is best done in a way that is object-oriented, incremental, evolutionary, and co-operative. *Object-oriented* programming and programming languages (Madsen et al., 1993) help ensure that the system being built is among others robust, correct, and maintainable. *Incremental and evolutionary* development (Floyd, 1994) helps in ensuring that systems meet (emerging) requirements. *Co-operative* analysis, design, and implementation (Greenbaum et al., 1991), in which (end) users actively participate, ensures crucial input from and output to users. The participation should consider different levels of implementing the system, ranging from technical to organisational.

Then, tools, techniques, and approaches that support an object-oriented, incremental, evolutionary, and co-operative process become crucial. This position paper is concerned with tools and techniques helping in doing this by supporting a certain mix of competencies, namely that of (OO) developers and (end) users.

Since model building is a cardinal point in system development, support for this becomes important. The UML (Rumbaugh et al., 1999) has emerged as a standard for much modelling. However, the UML is not usable by end users. This means that, in collaborative work settings, formal and informal modelling needs to be mixed. Basically, two types of technology exist for doing object-oriented modelling, namely blackboards and (CASE) tools. Use of blackboards are informal, supports synchronous collaboration, and enables extensions of notations. CASE tools, on the other hand, combine a formal notation with the possibility for asynchronous collaboration and tool integration. To facilitate effective user collaboration in modelling activities, these two technologies need to be combined. The Knight Project aims at doing this.

2 Insights from User Studies

We have made user studies (ref) of different user groups. The groups can be characterized along two dimensions: End users vs. developers and experts vs. novices. All of the different possible constellations in this space with respect to collaboration would be interesting to consider and study. However, so far we have mainly focussed on three of these. These studies will be discussed individually below, followed by a discussion of the key insights from these studies.

2.1 Experienced Developers and End User Domain Experts

This study has been carried out, informally, during the Dragon Project (Christensen et al., 1998). The project involved participants from a university research group and a major shipping company. Development took place in active collaboration with users in the sense that at least one user would be co-located with the development group at any given time. Whenever modelling of (conceptually) major areas of the shipping domain took place the users would participate actively in this. Actual modelling in these sessions would almost always take place on a blackboard. Although most information delivered by the users was in the form of domain knowledge as verbal or written accounts, drawings would also often be made by the end user on or in connection to the image of the model on the blackboard. User drawings would be informal, in contrast to formal UML models, describing key concepts or relationships from the shipping domain. Eventually, the users would nevertheless use parts of the UML notation and comment on e.g. multiplicities on an association.

2.2 Experienced Developers and Domain Experts that are Inexperienced Developers

We have studied a technology transfer project involving a university research group and an industrial partner (Centre for Object Technology, <http://www.cit.dk/COT>). The project reengineered an existing industrial application, while the inexperienced developers from the industrial partner learned object-oriented analysis and design. Modelling efforts in this project would take place using a mixture of CASE tools, projectors, and blackboards. Simplistically speaking, the inexperienced developers would explain the original implementation, whereas the experienced developers would model a reengineered version. Eventually, this balance shifted as the inexperienced developers would pick up larger parts of the UML and participate in the modelling. This would, naturally, introduce a certain number of syntactical errors in the use of the UML. These errors were either repaired, if they were disturbing a common understanding of what was modelled, or ignored, if the meaning was clear from the context. Following modelling sessions photographs would be taken of the models on the blackboard for future record or for inputting into a CASE tool.

2.3 Inexperienced Developers and Domain Experts that are Experienced Developers

This study investigated a redesign of an integrated development environment (Mjølner, <http://www.mjolner.com>). The group performing the redesign consisted of six persons with different experience in the domain (i.e., the integrated tool). All developers had experience in object-orientation and a fair understanding of the UML. Two of the developers had an in-depth knowledge of the environment, other two developers had a knowledge of the part of the tool that he developed, and the last two developers were introduced to the software architecture of the environment while participating in the redesign. In the redesign session the most used artefacts were a blackboard and a laptop. The blackboard was used to draw up the software architecture of

the existing environment and to make changes to these drawings. The laptop was used whenever a developer needed to look at code in order to remember the actual architecture. This use could take place whenever another person was at the blackboard. Although almost everything they drew was in actual UML notation, the notation was tweaked in three ways. First, UML did not suffice to explain certain aspects of the architecture leading to informal drawings of this. Second, the language used to implement the environment is BETA. BETA has a number of language and modelling constructs not supported by the UML including inner and virtual classes. These constructs were used heavily in the implementation, and thus the developers had to invent new notational elements on the fly. Third, the information drawn was filtered in the sense that only important attributes, operations, and classes were shown.

2.4 Key Insights

From our analysis of the user studies a number of lessons on the collaborative, communicative, and coordinative aspects of object-oriented analysis and design can be learned. For the purpose of this position paper, we can group the insights into two categories on use of drawing and collaboration.

Use of Drawings

Most of the drawing elements made were in the form of UML diagrams, i.e., class, sequence, and use case diagrams. However, these elements were combined with non-UML elements in one of two forms. Either as rich "freehand" elements that explained part of the problem domain, or as formal additions to the UML such as notations for inner classes or grouping. This means that a tool supporting collaborative object-oriented analysis and design should, ideally, facilitate: drawing in a formal modelling notation, drawing of "freehand" additions, and the ability to tailor the notation "on the fly".

Another key observation is the use of filtering. Filtering was used for several reasons. First, even blackboard real estate is limited. Second, not all parts of a diagram are interesting at all times. Third, users may employ a specific semantic filtering to decide the important elements of a diagram. Such a filtering could e.g. be that only the first compartment of a class is shown, or that work is only done on the parts of an application considered with user interface.

In all user studies, drawing has been transferred to and from CASE tools. This naturally implies that the tool we develop should be, or integrate with, a CASE tool.

Also, a number of detailed observations of the actual interaction with the blackboard have been made. We have e.g. studied which elements of UML that were drawn, how they were drawn, and how and why they were edited.

Collaboration

It has been a striking fact in our user studies that all collaborative construction of models has been coordinated as turn-taking. This is somewhat in contrast to other observations on shared drawing (Stefik et al., 1987), but we believe it to be general for the kind of work that (object-oriented) analysis and design is all about.

What was however not coordinated via turn-taking, was verbal communication and use of other artefacts. The people engaged in the meetings would e.g. discuss among themselves while another person was drawing at the blackboard, or they would use other artefacts concurrently. This has design implications, especially if distributed collaboration is to be considered.

3 Design and Implementation of Knight

Based on the user studies, we have recently started developing a system, Knight, supporting collaborative object-oriented analysis and design.



Fig. 1. Situated, Collaborative Use of the Knight Prototype.

Since our own development process is evolutionary, incremental, and collaborative, this section can only report on our current implementation and prospective plans. All this can, and will, change.

The major enabling technology for the implementation of Knight, is a large touch sensitive computer screen. Currently, this is a Smart Board (Fig. 1., <http://www.smarttech.com>). The basic idea is that combining the Smart Board with gesture input (Rubine, 1999) should retain much of the ease of use that a blackboard provides. Classes and associations, e.g., can then be drawn with dry erase markers on the surface of the touch sensitive screen as boxes and lines respectively. Since the algorithm we use for gesture recognition makes it possible to incrementally train the recogniser, it will be possible to incrementally change the set of gestures.

One of the key observations was that users and developers tweak the notation by filtering in different ways. We support this by combining a notion of (semantically) zoomable interfaces with fish-eye views (Bederson et al., 1994). This will provide the users with a way of zooming while keeping peripheral awareness of the whole model, and of changing model elements while zooming according to specified criteria.

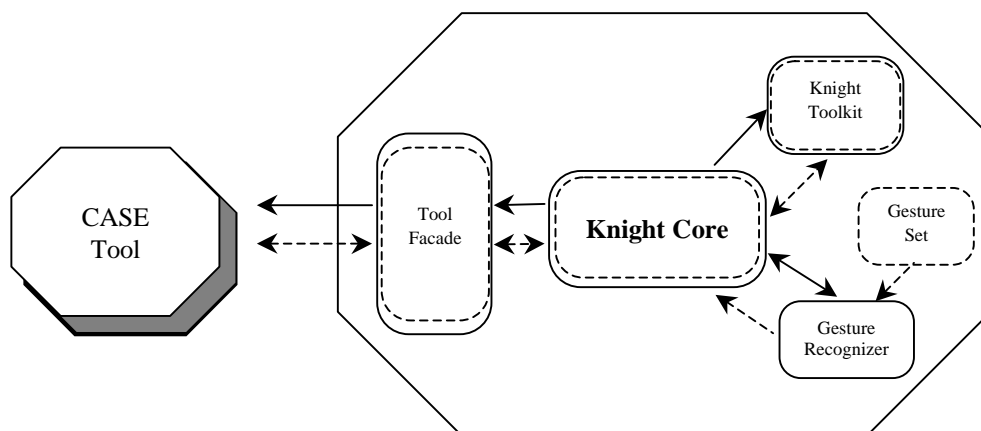


Fig. 2. Current, Conceptual Architecture of Knight.

The established design criteria make an architecture as shown in Fig. 2 appropriate for Knight. Drawn in a slightly modified version of Bass et al.'s (1998) software architecture notation, a core Knight process and a set of CASE Tool processes are shown. Knight will integrate with a set of CASE tools through OLE communication on the Windows platform. The idiosyncrasies of interfaces to individual CASE tool will be hidden by a Tool Facade component that defines a common CASE tool interface. The gesture recognition (the algorithm and the interaction issues) is crucial to Knight. This is shown by the two components implementing gesture recognition. Finally, in order to communicate with the Smart Board and to implement the zoomable interface an extended toolkit on top of the standard Windows toolkit is needed.

4 Status, Future Work, Implications, Conclusions etc.

We have looked at modelling, especially using UML, from a perspective of concurrency, collaboration, and communication. We have found that UML, mixed with less formal elements, may consist a base on which to build a tool supporting collaborative object-oriented analysis and design. We are currently developing such a tool.

5 Acknowledgements

Other than the author of this position paper, participants of the Knight Project team are: Christian Heide Damm, Michael Thomsen, and Michael Tyrsted. The work described in this position paper is partly supported by the Danish National Centre for IT-Research (CIT, <http://www.cit.dk>).

6 References

- Bass, L., Clements, P., Kazman, R. (1998) *Software Architecture in Practice*. Addison Wesley Longman.
- Bederson, B. B., Hollan, J.D. (1994) Pad++: A Zooming Graphical Interface for Exploring Alternate Physics. In Proceeding of User Interface Software Technology (UIST) 94.
- Bly, S.A., Minnerman, S. (1990) *Commune. A Shared Drawing Surface*. In SIGOIS Bulletin, Massachusetts.
- Christensen, M., Crabtree, A., Damm, C.H., Hansen, K.M., Madsen, O.L., Marquardsen, P., Mogensen, P., Sandvad, E., Sloth, L., Thomsen, M. (1998) *The M.A.D. Experience: Multiperspective Application Development in evolutionary prototyping*. In *Proceedings of the 12th European Conference on Object-Oriented Programming (ECOOP '98)*, Brussels, Belgium.
- Greenbaum, J., Kyng, M. (1991) *Design at Work: Cooperative Design of Computer Systems*, Hillsdale New Jersey: Lawrence Erlbaum Associates.
- Floyd, C. (1994) *A Systematic Look of Prototyping*. In R. Budde, K. Kuhlenkamp, L. Mathiassen, & H. Züllighoven (eds.), *Approaches to Prototyping*. Berlin: Springer Verlag.
- Madsen, O.L., Møller-Pedersen, B., Nygaard, K. (1993) *Object-Oriented Programming in the BETA Programming Language*, ACM Press, Addison Wesley.
- Rubine, D. (1991) *Specifying Gestures by Example*. In ACM SIGGRAPH Computer Graphics, 25(4), July.
- Rumbaugh, J., Jacobson, I., Booch, G. (1999) *The Unified Modeling Language Reference Manual*. Addison Wesley.
- Tang, J. (1989) *Listing, Drawing, and Gesturing in Design: A Study of the Use of Shared Workspaces by Design Teams*. Ph.D. Thesis. Stanford University, Department of Mechanical Engineering.