Benefits of Interactive Display Environments in the Software Development Process

Mathias Frisch
Otto-von-Guericke-Universität Magdeburg
Computational Visualistics/Software Engineering Group
D-39106 Magdeburg, Germany
+49 (391) 6711430
mfrisch@isg.cs.uni-magdeburg.de

Raimund Dachselt
Otto-von-Guericke-Universität Magdeburg
Computational Visualistics/Software Engineering Group
D-39106 Magdeburg, Germany
+49 (391) 6718772
dachselt@isg.cs.uni-magdeburg.de

ABSTRACT
Models become increasingly important for software development processes. Though there is a multitude of software modeling tools available, the handling of diagrams is still difficult. To overcome these problems we propose the usage of novel visualization and interaction techniques for the software development process, including multi-touch displays, the integration of diagrams drawn by hand and the interaction through zoomable user interfaces.

Categories and Subject Descriptors
H.5.2 [**Information interfaces and presentation**]: User Interfaces – *Graphical User Interfaces*
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *User Interfaces*

General Terms
Human Factors

Keywords
Software development process, UML, models, diagrams, interaction techniques, visualization, multi touch, zoomable user interface, semantic zooming

1. INTRODUCTION
A multitude of studies has been conducted to understand typical activities, needs and behavior of people involved in software development processes. They focus on different aspects like co-located meetings [5], the needs of distributed teams [7] or the usage of diagrams [4]. Some of the main requirements for tool support determined by these studies are:

- Providing different views for different people to the same content would be beneficial, e.g., a high level view for customers or managers and a low level system view for developers.
- Digitalization of handwritten diagrams and their integration in models or documents should be quick and easy.
- Collaborative work should be supported.
- A complete overview of the system and the visualization of dependencies between components should be possible.

Dealing with diagrams is characteristic for all these requirements. On the one hand diagrams may be hand drawings with an informal and transient character [4]. On the other hand they can represent very formal, complex and detailed models which become more important by model driven approaches. Both types have to be considered.

Current software modeling tools (e.g., [17], [14]) do not offer sufficient support for those needs. They implement logical interrelationships and dependencies but don't visualize them. Navigation in diagrams is often too cumbersome and a simultaneous view on macroscopic and microscopic levels of detail or a smooth transition between them is not possible. Beyond that they hardly support collaborative work.

In this position paper we propose to use the potential of interactive displays of different size (from huge wall sized displays to tiny handhelds) for the domain of software development. This includes multiple devices which collaborate with each other as well as novel interaction and visualization techniques which promise an easy and intuitive way of interaction. Multi-touch displays, where several people can interact simultaneously, and zoomable user interfaces are just two examples.

Recently, some of these techniques have found their way to home environment and entertainment applications. For instance, devices like iPhone or iPodTouch are equipped with multi-touch displays and game consoles like Wii offer new interactive possibilities. Other hardware such as multi-touch screens (e.g., [11], [13]) is not yet widespread and hardly available for consumers but this might change in the future. It is expected that people will get used to the convenient ways of interaction offered by these devices. To realize such equipment new interfaces were developed and studies were carried out to examine different kinds of applications, also in multi user settings (e.g., [9], [10], [12]). However, to our knowledge, up to now there are no analyses dealing with these novel interaction paradigms in software engineering and concrete software development processes.

CHASE'08, May 13, 2008, Leipzig, Germany.

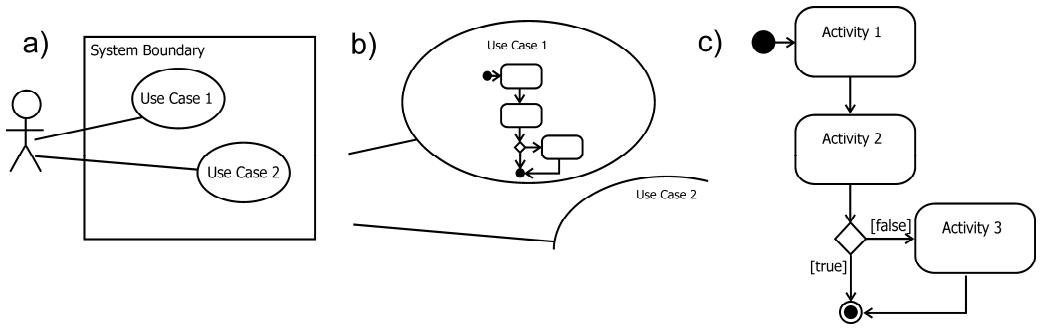

Figure 1. Three levels of detail while zooming in on a use case, (a) UML use case diagram, (b) zoom on Use Case 1 shows preview of activity diagram which describes the process in the use case, (c) activity diagram with all features on the most detailed level

This position paper is structured as follows: Section 2 describes interaction and visualization techniques which might be applicable in the software development process. We also address problems and challenges which might come up concerning the application of these techniques in the domain of software engineering. Section 3 suggests scenarios where novel mixed display solutions can be used and the presented technologies can be applied to the software development process. This section should provide a basis for discussions in the workshop. In section 4 we describe what we plan to do in the future.

2. Novel Interaction and Visualization Techniques applied to Software Development

2.1 Animated Diagram-Transitions

2.1.1 Transitions between diagrams

In many cases the same artifact (e.g., a certain class) appears in several diagrams, which means that these diagrams are connected to each other on a logical level. Visualizing this kind of connections could improve the overview and the understanding of the whole model. To realize this, we suggest a smooth animated transition between diagrams using a specific artifact as a pivot point. Blending between diagrams is beneficial for reducing the cognitive load while switching diagram types. This approach could be helpful when, for example, an instance of a class is shown in a sequence diagram and the developer wants to know in which other diagrams this instance appears or where the appropriate class is defined.

2.1.2 Semantic Zooming

Another problem is to provide a smooth transition between a coarse overview-model to a more detailed one. Current software modeling tools implement geometric zooming to scale diagrams up and down. We suggest the usage of semantic zooming techniques where the appearance of the focused artifact or group of artifacts changes from level to level. A prototype which implements semantic zooming on UML-Diagrams was introduced by [8]. This tool implements the zooming functionality for package and class diagrams including a focus and context view. On coarse levels package diagrams are shown. Zooming in on one of their components makes the classes inside the package visible. However, for other types of models such as behavior or interaction diagrams zooming is not yet available.

There are several situations where semantic zooming on such models would be beneficial, e.g., a zoom on use cases could end up in a sequence or an activity diagram. The latter show more details of the process within the use case (see Figure 1) and provide information such as if the corresponding requirement was considered and how it has to be implemented.

This raises the question for which types of diagrams it makes sense to use semantic zooming, or in other words which diagrams can be "nested" into each other in a sensible way. Another problem with respect to this kind of interaction is the diagram layout. If on a more detailed level some parts are changed (e.g., a class is added) this can also have consequences for the alignment of components in coarser levels.

2.2 Multi-touch displays

Multi-touch screens allow several users to interact simultaneously with content by direct manipulation. Interaction can take place by means of finger gestures, digital pens or even tangible widgets. These techniques are rather natural and can therefore lower the training effort for inexperienced users. A multitude of techniques exists to ease interaction with multi-touch displays. We expect that some of the problems addressed by these solutions also apply to the domain of software development.

During collaborative work on huge displays the quick accessibility of screen content far away from the user is a challenge. Approaches like [3] and [15] were developed to solve this issue. Concerning software development, they could be used to add certain artifacts to existing models. Multi-user techniques like *Storagebins* [16] and *Currents* [6] could be used in early stages of the development process like brainstorming sessions, collecting requirements and identifying use cases or classes, for instance with Class-Responsibility-Collaboration Cards. Some techniques deal with the handling of piles of artifacts on multi-touch displays (e.g., [1]). They could be potentially used to collapse and expand parts of class or use case diagrams. As far as we know, none of the mentioned interaction techniques have been tested in the domain of software development so far.

2.3 Digital pens

Digital pens make the quick digitalization of hand drawings possible. One example is the Anoto functionality [2]. Digital pens enabling Anoto functionality contain an integrated digital camera. It takes snapshots of a dot pattern printed on the paper and almost invisible to the human eye. The data collected this way is sufficient to determine the exact position of the pen and what it writes or draws (see Figure 2). This approach could be used to digitize sketched diagrams and to integrate these, e.g., in existing

digital models just by putting the drawing on an interactive tabletop display where it is recognized by the system. A digital copy of the drawn content could be downloaded from the stylus, could appear on the display, and could be easily integrated into existing models, e.g., by hand gestures.

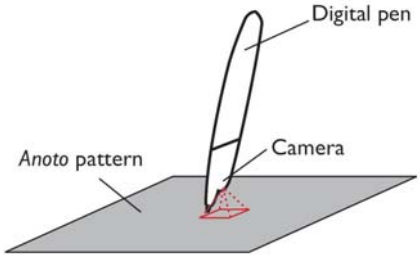

Figure 2. Functionality of Anoto [2]

3. Example Scenarios

In this section we will present example scenarios to illustrate the utilization of multiple interactive displays in the software development process. The aim is to provide a general basis for discussion.

3.1 Single User Scenario

In a single user scenario a software developer uses his own workspace (which can be a common PC) to edit diagrams or to program. This gives him access to the parts of the system which are currently interesting for him.

Additionally installed interactive displays can show diagrams which give an overview of the whole architecture. A zoomable user interface and animated transitions between diagrams can accomplish a quick navigation between macroscopic overview and microscopic detail view or between different parts of the architecture respectively, as described above. Beyond that, currently edited parts can be simultaneously visualized, e.g., recently edited methods are highlighted in the diagram.

We also envision a combination of several interactive multi-touch displays. Two of these displays could be installed orthogonally to each other to show multiple diagrams simultaneously. In this way an overview diagram on one display and a more detailed visualization of the same content on the other one would be possible. A further application would be to visualize logical relationships. As mentioned before, it is feasible, for instance, to see where certain components of a diagram appear in other diagrams. The selection of a class on screen 1 could cause the indication of an instance of the same class in a corresponding sequence diagram on screen 2. For that, we need to investigate appropriate layout options. As shown in Figure 3 the relevant parts of the diagram can be placed, for example, at the connecting edge of the two displays to visualize their interconnection.

This approach could help to become acquainted with an unfamiliar system more quickly, because of a better overview and multiple different views on static and dynamic aspects. Another advantage could be that dependencies between certain parts of the system might be easier to detected, especially when divers people working on the development of a system and edit different parts of it separately.

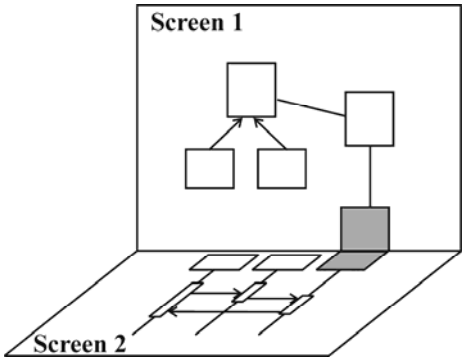

Figure 3. Sketch of two orthogonal displays, screen 1 shows a class diagram, an instance of one of the classes appears in the sequence diagram visualized on screen 2

3.2 Multi User Scenario

Many software development activities take place in meetings. Communicating and solving problems, designing systems or introducing new team members are just some examples. The number of participants can vary. It is also possible that not all attendees are in situ, but take part via remote connection.

Horizontally and vertically arranged interactive displays can be used as digital canvases to visualize diagrams or notes. In [5] a summary of requirements for conference support software for software design meetings is given. We think that many of these requirements could be solved with the mentioned technologies. It is possible to split the display and dedicate certain areas to certain people. By means of semantic zooming it is possible to hide canvases in other canvases and to rescale content conveniently. This technique allows also a quick change from a coarse system-overview for customers to a more detailed one for software engineers in an easily comprehensible way. It is also conceivable to order artifacts by certain (automatically collected) metadata like time, place, author etc. to ensure quick searching and finding. Especially in this scenario the mentioned Anoto functionality could be applied to combine the advantages of quick sketches on paper and digital diagrams. Important hand drawings do not have to be digitalized after the meeting in a cumbersome and error-prone way and the flexibility of paper is still given.

Smaller portable devices such as TabletPCs or handhelds can function as private workspaces with an option to combine them with other devices. Approaches like the one introduced in [18] where two interactive tablets can be combined to a single display might be promising, too. Beyond that, mobile devices can serve as remote controls, not only by pushing buttons on the device itself, but also by moving it around and with motion recognition through acceleration sensors.

4. Future Work

Plans for our future work include the transfer of existing visualization and interaction techniques into the software engineering domain and the development of new interaction techniques respectively. In addition, we want to investigate how existing collaborative workspace systems are suitable for supporting software engineering activities and how the techniques presented in this paper can be integrated in these systems. We plan to implement promising approaches as prototypes and to evaluate them. First of all we will focus on the development of a

semantic zooming user interface for UML diagrams. In addition, we want to test this solution against state-of-the-art software modeling tools concerning user performance and feeling of overview.

5. Conclusions

In this position paper we presented our vision how new interaction and visualization approaches could improve the software development process. We presented different techniques, like multi-touch displays and zoomable user interfaces and stated their advantages. Beyond that we suggested sample scenarios where these approaches could be used.

6. REFERENCES

[1] Aliakseyeu, D., Subramanian, S., Lucero, A., and Gutwin, C. 2006. Interacting with piles of artifacts on digital tables. In *Proceedings of the Working Conference on Advanced Visual interfaces* (Venezia, Italy, May 23 - 26, 2006). AVI '06. ACM, New York, NY, 159-162.

[2] Anoto functionality, http://www.anoto.com/

[3] Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P. Bederson, B., and Zierlinger, A. Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch- and Pen-operated Systems. In *Proceedings of Interact 2003*, Zurich Switzerland, August 2003, pp. 57-64.

[4] Cherubini, M., Venolia, G., DeLine, R., and Ko, A. J. 2007. Let's go to the whiteboard: how and why software developers use drawings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA, April 28 - May 03, 2007). CHI '07.

[5] Dekel, U. 2005. Supporting distributed software design meetings: what can we learn from co-located meetings?. In *Proceedings of the 2005 Workshop on Human and Social Factors of Software Engineering* (St. Louis, Missouri, May 16 - 16, 2005).

[6] Hinrichs, U., Carpendale, S., and Scott, S. D. 2006. Evaluating the effects of fluid interface components on tabletop collaboration. In *Proceedings of the Working Conference on Advanced Visual interfaces* (Venezia, Italy, May 23 - 26, 2006). AVI '06. ACM, New York, NY, 27-34.

[7] Ko, A. J., DeLine, R., and Venolia, G. 2007. Information Needs in Collocated Software Development Teams. In *Proceedings of the 29th international Conference on Software Engineering* (May 20 - 26, 2007). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 344-353.

[8] Köth, O. and Minas, M. 2002. Structure, Abstraction, and Direct Manipulation in Diagram Editors. In *Proceedings of the Second international Conference on Diagrammatic Representation and inference* (April 18 - 20, 2002). M. Hegarty, B. Meyer, and N. H. Narayanan, Eds. Lecture Notes In Computer Science, vol. 2317.

[9] Mazalek A; Davenport G ; Reynolds M; Sharing and Browsing Media on a Digital Tabletop; IEEE Multimedia, Special Issue on Continuous Archival and Retrieval of Personal Experiences, 2006.

[10] Morris, M. R., Paepcke, A., Winograd, T., and Stamberger, J. 2006. TeamTag: exploring centralized versus replicated controls for co-located tabletop groupware. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Montréal, Québec, Canada, April 22 - 27, 2006).

[11] Microsoft Surface, multi-touch display, http://www.microsoft.com/surface/

[12] Nacenta, M. A., Pinelle, D., Stuckel, D., and Gutwin, C. 2007. The effects of interaction technique on coordination in tabletop groupware. In *Proceedings of Graphics interface 2007* (Montreal, Canada, May 28 - 30, 2007).

[13] Perspective Pixel, Inc., multi-touch displays, http://www.perceptivepixel.com/

[14] IBM, Rational Rose, http://www-306.ibm.com/software/awdtools/developer/rose/modeler/

[15] Reetz, A., Gutwin, C., Stach, T., Nacenta, M., and Subramanian, S. 2006. Superflick: a natural and efficient technique for long-distance object placement on digital tables. In *Proceedings of Graphics interface 2006* (Quebec, Canada, June 07 - 09, 2006).

[16] Scott, S. D., Carpendale, M. S., and Habelski, S. 2005. Storage Bins: Mobile Storage for Collaborative Tabletop Displays. *IEEE Comput. Graph. Appl.* 25, 4 (Jul. 2005), 58-65.

[17] Sparx Systems, Enterprise Architect, http://www.sparxsystems.de/

[18] Tandler, P., Prante, T., Müller-Tomfelde, C., Streitz, N., and Steinmetz, R. 2001. Connectables: dynamic coupling of displays for the flexible creation of shared workspaces. In *Proceedings of the 14th Annual ACM Symposium on User interface Software and Technology* (Orlando, Florida, November 11 - 14, 2001).