COSMOS: Multi-touch support for collaboration in user-centered projects

Daniel Löffelholz, Torben Pergande, Hauke Wittern, Olaf Zukunft

HAW Hamburg, Department of Computer Science
Berliner Tor 7, D-20099 Hamburg, Germany
<firstname.lastname>@haw-hamburg.de

Abstract: First multi-touch tables like the Microsoft Surface are commercially available. Their use allows to enhance the user experience in a large number of collaborative domains. In this paper, we examine how early phases in software development like requirements engineering can be supported by this new technology. We present a novel approach for supporting collaborative work between systems designers and stakeholders in a software development environment. Based on an adapted model of interaction we present COSMOS (COllaborative Surface for MOdeling Software), a framework for easy creation of tools that support collaboration between system designers and customers. We used COSMOS to implement a UML diagram editor and are currently implementing a business process editor on the Microsoft Surface platform. First experiences with the framework and the editor show that they can strongly support the collaboration between system designers and stakeholders.

1 Introduction

Collaborative activities in early software development phases still often rely on traditional media like whiteboards and paper sheets despite the ubiquitous computer technology. This tends to result in media discontinuity, redundant information, and unnecessary tasks like preparation of media and transcription of changes or annotations back to the digital media. Useful activities like on demand hiding of unused details or searching in larger documents is obviously not possible. Nevertheless most digital technology is not well suited for a meeting-like setting. E.g. projectors only support one conductor to control the application with a single input device, which is probably suited in a teaching environment, but not beneficial for profitable discussions or reviews. So amongst many tasks the benefits from face-to-face collaboration cannot be compensated by the use of digital technology. For this reason almost every work environment features one or more tables for enabling a face-to-face collaboration amongst co-located people.

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4

The upcoming interactive tables like – but not limited to – Microsoft Surface [MC11] now offer the unique possibility to combine several benefits from these two working styles. The available mature multi-touch technology avoids a complicated handout or preparation of input devices and offers a seamless interaction between small groups of people. However, gestural interfaces demand the development of new, unique and richer design approaches and patterns [SI01]. This is especially true if combined with the approach of "several people working simultaneously on a single display".

This paper examines how software development can benefit from the possibilities of these new interface characteristics in early design stages of a software development project. To present COSMOS (COllaborative Surface for MOdeling Software), our framework for developing modeling applications for interactive tables, this paper is organized as follows: After this introduction, we explain how multi-touch tables can support collaborative phases in software development. We discuss the limitations and benefits from the use of this new kind of technology. Afterwards, we highlight some differences concerning interface design and working style between the traditional model of computer supported work and the computer supported collaborative co-located work. As a case study, we present a first application build on top of COSMOS: An application for reviewing and adjusting class diagrams in the Unified Modeling Language (UML) on the Microsoft Surface multi-touch table. Finally, we summarize implementation aspects, give our conclusions and an outlook to future work.

2 Related Work

The fundamental benefits of tabletop computers compared to traditional computers have been summed up by Shen et al. [SRFE06] as follows: first, touch input reduces the cognitive load of users and allows intuitive manipulations. Second, the opportunity to gather around tabletop computers enables face-to-face interactions. This is important in collaborative environments. However, Shen et al. [SRFE06] admit that positive influences of tabletop computers on helping users to accomplish their tasks are not proven yet.

Subsequent studies indicate advantages of tabletop computers when accomplishing tasks. But the scenarios that were investigated in these studies are of synthetic nature and not related to modeling software. For instance, TeamTag [MPWS06] is an application for collectively annotating photos. These studies investigate in relevant common issues of tabletop computing, though.

One issue is the individual perspective of the users gathering around a tabletop computer. This issue is often addressed by allowing users to rotate the user interface or single elements of it [HCVW06, MPWS06]. However, Morris et al. [MPWS06] observed that that orientation has only low negative impact on the usability. Similarly, according to a study Wigdor and Balakrishnan [WB05] orientation has an effect on speed of reading text, but presenting text at non-optimal orientation does not severely impair reading performance. Therefore, they suggest not to maximize text readability at all costs. Designing an interface without duplicated labels saves screen space [WB05].

ISBN 978-3-88579-286-4

Another relevant research topic is whether to offer the users centralized or replicated controls. Morris et al. [MPWS06] found in their comparative study that users prefer replicated controls. Replicated controls require additional screen space, but alleviate the users' aversion to use these controls. Thus, Morris et al. recommend to create one copy of each frequently used control per user.

3 A model for interaction

In early software development phases the people involved in projects - developers and customers - usually come together in face-to-face meetings to discuss requirements and models of the system. These meetings can be supported by electronic devices with some advantages over working with pen and paper or blackboards. Interactive tables can support those meetings by presenting and supporting the manipulation of previously created artifacts, e.g. graphical diagrams that are traditionally used to discuss system aspects. Typical benefits of computer-based support are the persistence of results, the possibility of performing instantaneous simulation and code generation. This results in productivity gains by reducing the effort of storing and organizing paper sheets. We have built COSMOS to support these and new innovative features focusing on support for a collaborative working style.

Models play a central role in software development and there are several scenarios how they are used in meetings:

- models can be developed from scratch and edited,
- models can be reviewed and discussed,
- alternative designs might be compared,
- the results of static analysis on models might be discussed and
- prototypes generated from models can be discussed

The scenario we have chosen for the first prototype based on COSMOS are reviews of UML class diagrams.

3.1 Visionary Scenario

Dave is a software developer and supervisor of a five man division for developing software in a supply chain context. Dave and his team are assigned to a new project. They have to develop a new logistic system for one of their customers. Dave has just received the first requirements have just arrived to Dave, so he blocks a conference room with a multi touch table inside for the beginning of the next week. By then any member in his team has to read the requirements and reflect about some ideas.

ISBN 978-3-88579-286-4

At Monday morning, the meeting starts and every project member sits around a multi touch table, on which a blank screen is displayed. As supervisor and implicit moderator Dave opens the meeting, summarizing the project aims and basic requirements. The goal for this session is a first design brainstorming based on the information they got. Then the multi-touch table comes into play. Dave puts a figure looking like a toolbox onto the screen and under the figure a menu is shown, where the user can choose an UML diagram Type to create. After creating a new class diagram, Dave encourages the team to brainstorm about the static architecture design of the logistic software, they have to build. Creating a new class, by dragging a class template out of the toolbox onto the diagram, Dave starts the session. Then he starts editing the class by tabbing on it and putting in information by using the on-screen keyboard. Afterwards Dave discusses the first idea of this class with the group. After a short period of time, all of the project members are interacting with each other and the multi touch table. Simultaneous input by the participants is no problem because of the multi-touch enabled user interface. Sara and Keith simultaneously create new classes and afterwards edit them while Mia is connecting the classes with a new association without disturbing each other. At the end of the brainstorming a first model of the static domain-oriented architecture realized through a UML class diagram with connections etc. is created. Dave saves the model in an interchangeable format and sends the result to the involved parties.

One week later, Dave and his team finished the first design. Now Dave needs to evaluate the design with the customer. Therefore, Dave arranges a meeting with Sara and three representatives of the customer. The meeting is located in the same conference room with the multi touch table. After starting the meeting, Dave presents the domain-oriented design concept on the multi touch table and asks for feedback. Sara has the role of a moderator and shows the basic elements of editing the elements and invites everybody to interact with the model. Since the meeting is intended as a design review Sara animates the other participants to interact directly with the displayed diagram. During the meeting the participants discuss possible design changes, instantly apply them and discuss the resulting modified design. After the meeting, the feedback of the customers is directly merged in the design and the first milestone is reached.

3.2 Collaborative work or “Multi-Touch vs. Single-Touch”

As noted in the introduction, Personal computers, Notebooks, Tablet Computers or electronic whiteboards are other devices that can support such scenario, but they are not really intended for such collaborative meetings. These devices either require sitting very close to each other when using a small computer display or require sitting in front of a wall when using a video projector or a whiteboard. This might result in an uncomfortable atmosphere. The atmosphere during review meetings is already critical. The author of a reviewed document might get in a psychological strained situation when he feels himself personally attacked. Also a strict moderator role together with only one input device is not well suited for a collaborative work.

ISBN 978-3-88579-286-4

Multi-touch tables however, incorporate the advantages of electronic devices and further contribute to an easy-going atmosphere. They enable multiple participants of a meeting to simultaneously see all content on the display while sitting in a usual and comfortable setting. Multi-touch tables help to ease the potentially strained atmosphere. Our observations show that users focus considerably stronger on the content presented by any multi-touch table applications than by traditional applications. Therefore, we belief that the authors of reviewed documents stay in the background and are less often personally attacked when using a multi-touch table environment.

Furthermore, multi-touch tables do not imply the necessity of a "superuser" operating the keyboard and mouse but allow all participants to equally use the multi-touch table.

4 COSMOS: A Framework for creating interaction supporting tools

COSMOS is our newly created and implemented framework for the creation of tools that focus on supporting interaction.

4.1 Requirements

COSMOS is designed to solve a number of problems common to interaction supporting tools. First, since typical application scenarios like the one presented in section 2.1 require the use of different modeling notations, we decided not to implement a single modeling language but to provide a means for the creation of these domain-specific tools.

With COSMOS developers should be able to easily create editors for graphical domain specific languages. It shall support any concrete syntax that is based on graphs with nodes that can be connected by edges, e.g. different UML diagrams, Petri-nets and Entity-Relationship-diagrams.

Additionally, COSMOS should integrate support for information hiding in concrete models. To ease the understanding of models, details should be made visible or invisible depending on the context. Finally, COSMOS has to support a unique feature of the used technology: The limited space of the displays. Although we currently use multi-touch tables that are significantly larger than typical desktop monitors, the resolution on the table is significantly smaller than on the desktop. Hence, it is not feasible to reuse widgets of typical desktop applications.

4.2 Components of COSMOS

Any diagram editor that is created with COSMOS must define the supported domain, how the domain is displayed and how the domain can be edited. The application is configured using the following components:

ISBN 978-3-88579-286-4

1. a definition of the abstract syntax of the domain which defines the elements of the domain,
2. a static semantic which defines constraints against the abstract syntax,
3. a graphical concrete syntax for the abstract syntax which describes how the domain elements are displayed,
4. a set of either domain specific or unspecific tools that enable collaborative work with diagrams,
5. the application logic and
6. the application configuration.

The abstract syntax and the static semantic are a meta model of the domain. Together with the concrete syntax they define the Domain Specific Language (DSL) that is used by the editor.

A developer of a new editor must implement at least the application configuration. The application configuration sets up the viewing and editing capabilities as well as the available tools of the editor. The other components can be either reused components of the COSMOS framework or newly implemented components that are tailored for specific needs. Basic tools like the paper bin and common editing capabilities like creating new items from templates are included in the framework and can be reused for every graph-based editor. Figure1 shows the components of the UML editor.

4.3 Guiding thought process and abstraction mechanisms

A key for easy understanding of complex models are abstraction mechanisms and visual guidance. Therefore, COSMOS supports navigation between abstraction levels, leaving out unimportant details and visual effects that help transferring thoughts between stakeholders.

Typical to all graphical notations that are used for discussion are hierarchical compositions. COSMOS supports the nesting of diagrams into more abstract ones. This enables a smooth zoom transition between different abstraction levels, e.g. the nesting of UML class diagrams and UML component diagrams. Visually, the transition is animated as exploding a high level element to its nested lower level elements respective contracting nested elements into the containing element. This has shown to be a comprehensible metaphor and is similar to the semantic zooming technique presented by Frisch et al. [FD08, FDB08]. Any other state changes are also visualized with smooth animations. These animations avoid abruptly appearing or disappearing objects and help the user to understand what's going on [MC08]. Therefore, we consider animations not just as a nice to have feature but as a required piece of every application.

To enable the viewers to gain and keep an overview, COSMOS also supports the creation of different views for diagram elements. These views allow the user to switch between different degrees of detail. Switching between levels of details is illustrated by a flip-card metaphor with front and back side.

ISBN 978-3-88579-286-4

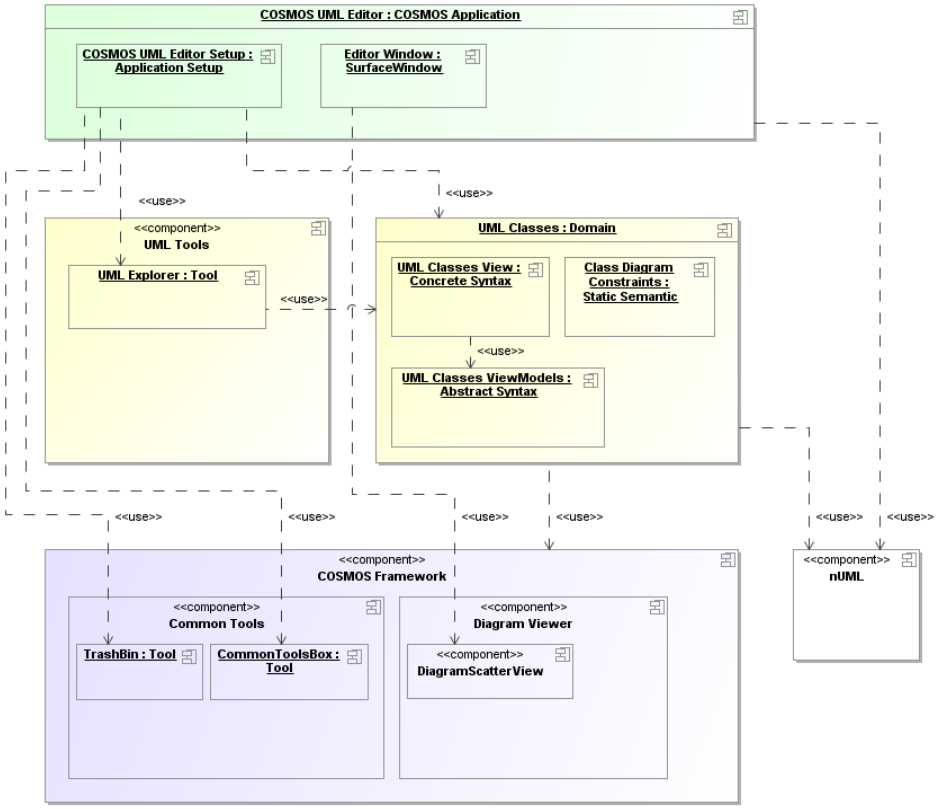

Figure 1: Components of the UML editor.

According to the survey by Cherubini et al. [CVDK07] relationships between diagram elements are usually represented with directed arrows. The diagram elements are arranged so that the arrows are generally pointed rightward or downward to indicate hierarchy. When sitting around a table it depends on where each observer sits whether an arrow is directed to the left, right, top or bottom. Therefore, the direction of an arrow and the arrangement of a diagram element might not have an intuitive meaning. COSMOS addresses this problem by enabling users to rotate the whole diagram. Hence everybody who is sitting around the table has the chance to watch the diagram with the correct orientation. However, further solutions must be found in the future.

ISBN 978-3-88579-286-4

4.4 Display Space Management

Another typical problem arises when designing a framework for multi-touch tables that is not found in traditional applications. Any multi-touch table provides only a limited amount of screen-space. The scrolling functionality of traditional graphical user interfaces does not work well in a multiuser setup. To minimize the space used to display the graphical elements, COSMOS offers different views for each element. The most relevant information like the element name can be displayed in an overview. When activated by a gesture more details can be displayed and edited. Thus, display space is occupied only when needed.

Finger contacts are larger and less precise than traditional mouse pointers. This has to be considered in the user interface design. We decided that input controls have larger interaction areas. To prevent unintended input, every input control must be accessible with at least one finger without touching any other control.

There are well known controls and metaphors on classical computation interfaces like context menus or paper bins. These are widely accepted and used by almost all users. In COSMOS, we first implemented similar icons on the table. Due to the limited display space described above, this has proved to be awkward. As an alternative, COSMOS uses another feature of multi-touch tables, namely the interaction with physical objects. The idea of using physical objects for interaction was first presented in [FIB95] where these objects are called graspable objects. The Microsoft Surface device is able to interact with any physical object placed on it. It recognizes the object by a visual tag stuck on it. This feature enables metaphors like placing a paper bin (see Figure 2 for a custom made bin) on the screen and only then displaying a paper bin tool inside the user interface. A diagram element can be deleted by dragging the virtual object into the physical paper bin as displayed in Figure 2.Whenever the bin is not used, it may be removed from the device and the display space is available for other purposes. Of course, the physical bin "contains" the virtual objects independent of its location. Generally, incorporating physical objects has the advantage of incorporating casually required functionality when needed. This functionality can be removed from the user interface by removing the object from the table. Thus, the very valuable resource "display space" can be saved. COSMOS currently uses this for the paper bin and the menu bar.

COSMOS does neither restrict how display space is used by the users nor requires placing controls in the center of the screen in order to make them accessible for every user. Using physical objects the number and the placement of controls is totally up to the users. Thus, the whole display is a shared space but personal spaces are allowed, too. Replicated physical objects satisfy the users' needs like replicated controls in [MPWS06]. But in contrast to those replicated controls they do only occupy screen space when needed.

ISBN 978-3-88579-286-4

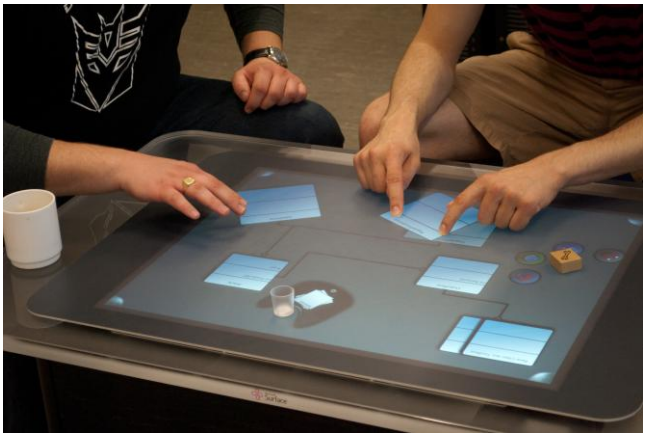

Figure 2: Using the UML editor based on COSMOS.

4.5 Reviewing and editing

The interaction model supported by our first prototype based on COSMOS is focused on reviews of models of a software system. During these reviews, the participants discuss the models to find discrepancies from planned results and to recommend improvements [STQ11]. The participants annotate the models with their findings but typically do not or only slightly modify them.

Our interaction model primarily concentrates on navigating through and between models as well as modifying how models are displayed to make it easier to understand the models as described in Section 3.2. Editing is a secondary interaction model that is also supported. We intend editing actions for demonstrating and discussing alternative designs. But we do not intend that participants of a review develop whole models from scratch with all necessary details. When creating new models, the kind of interaction is different from the kind when using existing models during reviews [MBH08]. Therefore, the development of whole models is not in the focus of COSMOS. Instead, we assume that a traditional modeling tool or a sketching tool like Calico [MBH08] is used to create the system description from scratch before we discuss it.

In a collaborative co-located work setting, where several participants interact with the same interface simultaneously or in short intervals, gestural and tangible interaction is an appropriate input technique. However, the lack of haptical feedback of touch-based interfaces as known from keyboard or mouse input devices needs to be dealt with. Therefore, in COSMOS every interaction is confirmed by optical feedback.

ISBN 978-3-88579-286-4

While traditional keyboard/mouse or even whiteboard interaction typically demands a single user to control the application (do things like typing or drawing), multi-touch interfaces do not need any handover or preparation of controls. Thus, the participants can seamlessly and spontaneously interact in a very democratic way. Users can rotate and resize every diagram-node as well as the whole application surface with habitual touch gestures, as they are used to rotate paper sheets and spreading cloth on a table respectively. Thus, the application is easily accessible from any position around the table. According to Wigdor et al. [WSFB07], this is important for multi-touch table applications.

With a single tap on a diagram element users can switch between the element's views. The views allow switching the level of details (see Section 3.2) as well as activating a clearly visible editing mode. In editing mode users can edit the properties of a node. In the domain of class diagrams such editable properties are the name, attributes or operations of a class. In cases where text input is required we use an on-screen keyboard. Thus, no additional input devices are required.

Editing of models is also supported with tools that can be enabled by placing an associated real world object on the table. The set of tools includes a paper bin for deleting and restoring elements by drag and drop operations. Model manipulations can further be undone or redone with the operation history tool. A tool with regular file operations like save and load is also included. This enables users to exchange models with external tools. For example UML models can be exchanged between regular UML tools via files in the XML Metadata Interchange (XMI) format.

5. Case study: UML diagrams on the surface

The first example application and proof of concept for the COSMOS framework is an implementation of an UML class diagram editor. We decided to start with UML class diagrams because they are widely known, accepted and can be used to discuss domain-oriented issues. Using COSMOS, UML class diagrams are implemented with a Textbox to visualize the name, a list for the attributes and another list for the methods of the class. Between any two classes, there can be three kinds of connections:

1. generalization,
2. aggregation, and
3. association.

This corresponds to the UML standard as defined by the OMG [OMG09]. A UML class has two views, namely an overview with class name, attribute names and type, and the method names. The other view is the editing mode, where all these elements of the class are editable via an onscreen keyboard. Only in this mode, the user is able to create connections between two objects by drawing a line from the diamond shapes at the sides towards the connectable class.

ISBN 978-3-88579-286-4

As an example, Figure 3 shows the screenshot of our UML class diagram editor built on top of COSMOS. There, the editor is used in the “review-mode”, where a customizable checklist is displayed. Each item on the checklist can be checked separately. The selected item interacts with the editor by displaying the required UML elements on the surface. In the middle of the top, you can see the toolbox with common functions like "new class" and "zoom out" to next abstraction layer, which were presented in 3.4. For UML class diagrams, we implemented a nesting into UML components. Placed on the buttom right is the paper bin, where the user can drop objects to be deleted. Like we stated above, both, toolbox and paper bin are only activated if the corresponding tagged figure is placed on the surface (in Figure 3 for technical reasons of the screenshot tool indicated only by small red rectangles. However, the physical objects are clearly visible in Figure 2).

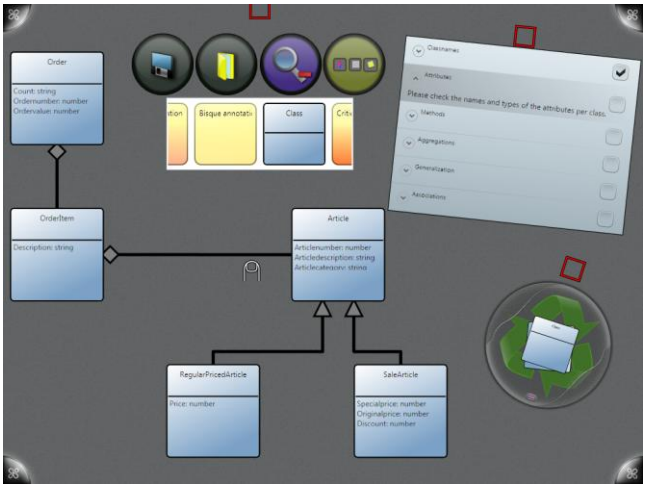

Figure 3: Screenshot of the UML editor based on COSMOS.

6. Implementation aspects

Unlike most mouse/keyboard applications, designing a multi-touch table interface is influenced by the type of underlying hardware, especially input and output technology. The Microsoft Surface device used by us deploys a DLP backprojection engine with a 1024x768 resolution, which makes it usable without any complicated setup of cameras. Additionally, it does not imply any restrictions for the room measures or lightning. The input recognition occurs through five infrared cameras, located in the underlying case. It can concurrently recognize and handle up to 52 input sources.

While developing the COSMOS framework and the UML class diagram implementation, three correlating modules were used.

ISBN 978-3-88579-286-4

1. The abstract syntax component can be created with any .Net programming language like C#. It specifies the object model of a diagram for a specific domain. For the UML editor, we used C# including the libraries as available in the Microsoft Surface SDK.
2. The concrete syntax component is mainly defined with XAML (Extensible Application Markup Language). XAML is a declarative XML-based language that is part of the .Net Framework. This component specifies how the elements (nodes and edges) of the domain are visualized.
3. The visual behavior of the elements can be further customized in any .Net programming language like C#.

Microsoft Surface specific controls, events and (finger-) contacts are part of the Microsoft Surface SDK. A specific Microsoft Surface control called ScatterView represents an area on which different objects are freely movable, rotatable and resizable. Because of the need for more functionality, e.g. the integration of a layout algorithm for connections, we decided to develop a new control named DiagramScatterView. Only this fits the requirements for COSMOS and is extensible for further features in the context of a framework.

Developing new applications, i.e. graphical editors with the COSMOS framework can be comfortably done using Visual Studio and Expression Blend. We found that using Expression Blend can significantly reduce the effort for developing the fragments that are implemented in XAML. Using Blend also resulted in a prettier look of the application. It allowed us to easily design graphical feedback to the user. For instance, applying glow effects on buttons, when they are touched, provides a great user experience.

7. Evaluation

During our iterative implementation process we continuously evaluated the usability of specific features of COSMOS at the end of each iteration. We also evaluated the usability of a stable prototype in detail. All participants of the evaluations were not members of the COSMOS-team but had profound background in software engineering.

At the end of each iteration, we mainly focused our evaluations on new or modified features. We observed the participants while they used COMOS and interviewed them after they finished some small tasks. Their feedback and ideas were systematically analyzed and influenced the subsequent iterations. This way we improved the features of COSMOS and found missing features that the users requested. Especially, we early got positive feedback about the physical objects. The users' requests for more such functionality strongly indicated they would accept them. On the other hand, the users often criticized multi-touch input. They complained that some multi-touch gestures were not intuitive. Only with the simple gestures rotate, move, scale and tap users had no problems. Other gestures were too complicated and hard to learn. Moreover, single-tap and double-tap gestures were in conflict and could be executed by mistake. Therefore, we decided to provide the single-tap gesture only.

ISBN 978-3-88579-286-4

Finally, we evaluated a version of COSMOS that included all important features in a larger usability test. The usability test was conducted in four test runs and two users per test run. Each test run simulated a scenario where two software developers review and to correct an existing UML class diagram. The given model included some defects that required the test-persons to alter text, to delete a superfluous class and to create a missing class.

The users' actions and conversations were recorded with three cameras. We requested the test-persons to think aloud so we could better understand their intentions and find possible usability issues. Additionally, we used a custom made contact-logger to record the multi-touch interactions. Although our lab featured an eye-tracker we did not use it. It emerged inappropriate for our scenario in which several persons could move freely around the table.

Summarized the results of our usability-test are:

- All test-persons had some initial problems to understand the handling of the application.
- All test-persons were able to overcome these problems within at most three interactions.
- All test-persons rated the application as an improvement to the discussion of models using a conventional WIMP-interface.

We consider the first drawback as acceptable since the test-persons had neither used a multi-touch application nor physical objects as input before. However, in reality at least one user should be familiar with the application. Our usability-test indicates that if a tabletop computer is used the inexperienced users quickly learn how to use the application because they see how the other users interact with the application.

8. Conclusion and future work

This paper introduced the framework COSMOS and a first application for a specific type of CSCW-applications: Using a multi-touch table for co-located discussion about and editing of UML diagrams. Due to the fact that collaborative work is still inadequately supported by traditional user interfaces, this paper examined several new and rich forms of interaction with an interactive table. We especially focused on space maximizing, supporting the collaborative work style, and intuitive interaction for a better user experience. Those design patterns can help developing a user interface which is not only usable by several people, but also enhances their collaboration which each other.

ISBN 978-3-88579-286-4

There are different aspects which need to be investigated in the scope of future work. Compared to the limited size of the currently available Microsoft surface, interaction on larger tables will need additional interaction forms e.g. for reaching distant elements. We still need to do a more detailed usability study on the impact of multi-touch table interfaces on working efficiency and user acceptance. Another open point is to enable a seamless data transfer between the multi-touch table and desktop computers as well as mobile devices. Currently, we rely on XMI for the UML tool. Finally, we are currently building a petri-net-editor and an editor for a storyboard on top of COSMOS and expect new insights into the features required from COSMOS and the usability of COSMOS for implementing new applications.

References

[CVDK07] Cherubini, M.; Venolia, G.; DeLine, R.; Ko, A. J.: Let's go to the whiteboard: how and why software developers use drawings. In: CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems. Pages 557-556. ACM Press 2007

[FD08] Frisch, M.; Dachselt, R.: Benefits of interactive display environments in the software development process. In: CHASE 08: Proc. of the 2008 Intl. Workshop on cooperative and human aspects of software engineering. Pages 53-56. ACM Press 2008.

[FIB95] Fitzmaurice, G. W.; Ishii, H; Buxton, W. A. S.: Bricks: laying the foundations for graspable user interfaces, In: CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems. Pages 442-449. ACM Press 1995.

[FDB08] Frisch, M.; Dachselt, R.;Brueckmann, T.: Towards seamless semantic zooming techniques for UML diagrams. In SoftVis 08: Proc. of the ACM Symposium on software visualization. Pages 207-208. ACM Press 2008

[HCVW06] Hancock Mark S.; Carpendale, Sheelagh; Vernier, Frederic D.; Wigdor, Daniel; Shen, Chia: Rotation and Translation Mechanisms for Tabletop Interaction. In Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP '06). IEEE Computer Society, Washington, DC, USA, 2006; Pages 79-88.

[MPWS06] Morris, M. R.; Paepcke, A.; Winograd, T.; Stamberger, J.: TeamTag: exploring centralized versus replicated controls for co-located tabletop groupware . In: CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems. Pages 1273-1282. ACM Press 2006

[MC11] Microsoft Corporation: Microsoft Surface, http://www.microsoft.com/surface

[MC08] Microsoft Corporation, "Microsoft Surface Interaction Design Guidelines", Oct-24-2008, URL: http://community.surface.com/downloads/p/156.aspx

[MBH08] Mangano, N. and Baker, A. and van der Hoek, A.: Calico: a prototype sketching tool for modeling in early design. In MiSE '08: Proceedings of the 2008 international workshop on Models in software engineering. Pages 63-68. ACM Press 2008.

[OMG09] Object Management Group (OMG), UML, 2009, http://www.uml.org/ .

[SI01] Shoemaker, G. B.D.; Inkpen, K. M.: Single display privacyware: augmenting public displays with private information, In: CHI '01: Proceedings of the SIGCHI

conference on Human factors in computing systems. Pages 522—529. ACM Press 2001.

[SRFE06] Shen, Chia and Ryall, Kathy and Forlines, Clifton and Esenther, Alan and Vernier, Frederic D. and Everitt, Katherine and Wu, Mike and Wigdor, Daniel and Morris, Meredith R. and Hancock, Mark and Tse, Edward: Informing the Design of Direct-Touch Tabletops. In: IEEE Comput. Graph. Appl. 26 (2006), No. 5, Pages 36–46.

[STQ11] Software Testing Qualifications Board (ISTQB): Standard glossary of terms used in Software Testing. 2011.

[WB05] Wigdor Daniel; Balakrishnan, Ravin: Empirical investigation into the effect of orientation on text readability in tabletop displays. In Proceedings of the ninth conference on European Conference on Computer Supported Cooperative Work (ECSCW'05), Hans Gellersen, Kjeld Schmidt, Michel Beaudouin-Lafon, and Wendy Mackay (Eds.). Springer-Verlag New York, Inc., New York, NY, USA, 2005; Pages 205-224.

[WSFB07] Wigdor, D; Shen, C.; Forlines, C.; Balakrishnan, R.: Perception of elementary graphical elements in tabletop and multi-surface environments. In CHI 07: Proc. of the SIGCHI Conference. Pages 473-482. ACM Press 2007.