EMIL: A Rapid Prototyping Authoring Environment for the Design of Interactive Surface Applications

Johannes Luderschmidt, Nadia Haubner, Simon Lehmann, and Ralf Dörner

RheinMain University, Wiesbaden, Germany
{johannes.luderschmidt,nadia.haubner,simon.lehmann,ralf.doerner}@hs-rm.de

Abstract. Interactive surfaces (IS) like digital tabletop systems offer a cornucopia of input possibilities like touch gestures or interaction with physical objects. Additionally, multiple users can interact simultaneously allowing for a collaborative setting. These aspects have increased the complexity of designing such interfaces as compared to WIMP interfaces. However, existing UI design approaches fall short of taking these aspects into account and existing design approaches for IS focus on software development. We introduce the EMIL environment that allows authors of design teams to create multi-touch and tangible user interfaces. In its core, EMIL consists of a software framework that provides interaction components (for instance, widgets like images or maps as well as interaction concepts like gestures) that are especially suited for IS. Authors like UI designers collaboratively create software prototypes directly at the IS without the need to write code. For this purpose, they use and adapt the components of the software framework in an authoring application. Authors collect and retrieve information about the interaction components in a knowledge database employing a tablet computer app. In a qualitative evaluation interview, EMIL has been well received by a design team of an advertising agency.

Keywords: interactive surfaces, multi-touch, tangible user interfaces, engineering of interactive systems.

1 Introduction

So far, research in the area of interactive surfaces has concentrated on hardware, gestural interaction and software frameworks. In contrast, approaches how typical design teams of a software company handle creating IS applications have not been researched well.

In this paper, we present the EMIL (**E**nvironment for **M**ulti-touch **I**n the **L**aboratory) environment that offers a rapid prototyping approach based on libraries and tools allowing design teams to collaboratively develop IS prototypes. In its core EMIL consists of a UI framework providing building blocks of IS applications. To be more precise, it employs specialized multi-touch and tangible user interface interaction components and application templates. However, simply providing a software framework would only satisfy developers' needs when

M. Kurosu (Ed.): Human-Computer Interaction, Part I, HCII 2013, LNCS 8004, pp. 381–390, 2013.

it comes to creating interactive surface software. Furthermore, programming IS applications on a desktop computer in an integrated development environment makes testing of interaction on the actual IS hardware difficult and is therefore performed infrequently [5]. Hence, our EMIL authoring application (EAA) allows collaboratively creating IS applications directly at the surface without the necessity to write code. Authors can as well use the components of the UI framework as build widgets easily themselves at the surface employing resources they imported in EAA. EMIL enables a prototyping process in which different kinds of authors (UI and interaction designers as well as programmers) form a design team and build the prototype in iterative, alternating design cycles. In one cycle, UI and interaction designers shape the concept of the prototype's UI and interaction and create graphical resources at their desktop computers employing their accustomed tools like Photoshop. Programmers adapt and enhance components of the UI framework according to designers' needs in their IDE. In the other cycle, the authors meet at the surface to build and refine the prototype. Created prototypes can be used for initial user tests, for client feedback and as a foundation for the actual application.

However, simply providing design tools would fall short of handling design knowledge that is necessary to build an IS prototype. Therefore, another part of the EMIL environment is the EMIL pattern authoring and browsing system (EPABS). EPABS constitutes a database of IS design knowledge. This database stores experience reports, user study results and examples of component's application fields in the form of interaction patterns [1]. Authors retrieve and enhance interaction patterns in the database employing the EPABS app on a tablet computer.

EMIL's component library, the authoring tool and application templates facilitate have the potential to speed up IS prototype creation. The knowledge browser allows systematically building and retrieving corporate IS software design knowledge. In an expert interview we qualitatively evaluated EMIL with a design team. In this interview, EMIL's basic concepts were received well.

This paper is structured as follows. In section 2, we present related work. We introduce the EMIL environment in section 3. Section 4 presents the results of the qualitative evaluation. Finally, section 5 gives a conclusion and presents future work.

2 Related Work

There exist several frameworks, toolkits and libraries which enable programmers to develop interactive surface applications. reacTIVision [4] provides a toolkit consisting of a computer vision tracking application and a network-based dialect to build IS hardware and software. TISCH [2] presents a similar approach but additionally introduces a so called widget layer with which multi-touch applications can be built. Pure software frameworks like PyMT [3] and MT4J [9] enable the creation of IS applications. The mentioned frameworks and toolkits offer powerful tools for system builders and programmers, but are less fitted for

UI or interface designers [6]. Therefore, in the following, we focus on concepts assisting programmers on the one hand and supporting visual development (for UI and interaction designers) on the other hand.

In [8], Landay and Myers present an interactive user-interface design tool that allows designers to build WIMP UI prototypes based on the recognition of electronic sketches. However, their concept is based on WIMP UIs and not on IS. The OpenInterface (OI) Framework [10] allows for a multimodal UI creation process that allows to include input channels into an OI component. Such a component can be used in OI's graphical authoring application SKEMMI to rapidly build multimodal interfaces based on a data flow graph. However, OI's focus lies on the creation of a flexible input channel architecture and not on the process of building the GUI itself.

Squidy [7] is a zoomable environment for the design of natural user interfaces (NUIs). Squidy differs from OI in that it addresses the issue that authors involved in the authoring process of NUIs often use multiple toolkits or frameworks to create the desired UI and its behavior. This is achieved by tying together relevant frameworks and toolkits in a common library while a visual language is introduced to create NUIs. To see the current development status as a whole, Squidy introduces the concept of semantic zooming, making it possible to control the level of complexity shown to the particular author developing the application. Squidy provides a way to take input data of hardware devices (e.g., movement data recorded by a Kinect), process the data and send it to listening applications, e.g., via Tuio [4]. The focus on technical aspects is important to develop the actual available NUI input alternatives for a specific application. Therefore, Squidy's focus is not on the creation of IS software in design teams.

The Designer's Augmented Reality Toolkit (DART) [12] is based on the multimedia development environment Director and allows authors to build augmented reality (AR) applications. It supports early design activities, especially a rapid transition from low-fidelity prototypes to working applications. Hence, DART allows to test prototypes early and often. The authors gained the experience that designers need to use their own tools for content creation. They provide so called behaviors that can be easily attached to content created by authors and can be easily extended if necessary. Although authoring experience made in the field of AR cannot be directly transferred to IS, DART's promising design approach that supports designers in the prototype creation can also be applied to IS design.

To sum up, the introduced visual prototyping approaches allow to rapidly build interface prototypes in the field of WIMP, post-WIMP and AR interfaces. However, interactive surfaces and their characteristics are not covered by this related work.

3 EMIL Environment

EMIL is an authoring environment for the creation of interactive surface prototypes. It supports design teams in their efforts to collaboratively build software for target platforms like multi-touch tables or touchable wall displays.

A design team in the sense of EMIL typically comprises designers creating the look and feel of the UI and programmers developing the actual code (see figure 1). To support such design teams, EMIL offers the EMIL authoring application (EAA) with which prototypes can be created out of a set of components provided by the EMIL UI framework. To inform the team about existing solutions and to store prototyping results, the EMIL Pattern Authoring and Browsing System (EPABS) offers a design knowledge database represented by interaction patterns that can be accessed with a tablet computer app.

Fig. 1. Three authors modifying a prototype in the EMIL authoring application on a tabletop system. In the foreground, a UI designer modifies the image of a map marker in Photoshop on his computer. As soon as the UI designer saves the Photoshop file on his laptop to the cloud storage, the changes become available in the prototype and all components that contain the marker resource from Photoshop can be updated.

Section 3.1 introduces the EMIL UI framework. In section 3.2, we present EAA. Finally, section 3.3 describes EPABS.

3.1 EMIL UI Framework

The EMIL UI framework provides visual and non-visual components for the prototyping process. Visual components can be widgets like lists, geographic maps, browsers, media like images and videos et cetera. Another kind of visual components are so called views. A prototype can contain several views which themselves contain widgets. A view navigation allows switching between views. Furthermore, application templates are a combination of prepared views and widgets. For instance, a consulting application template comprises specialized views and widgets tailored for user scenarios in which, for instance, a bank consultant wants to use an IS as support medium in a mortgage consultation of a customer.

Non-visual components are controls and behaviors. Controls provide gestural input to widgets like multi-touch transformation controls that allow dragging,

rotating and scaling based on standard gestures (see also [11]) and flick controls that allow for a momentum that keeps widgets moving after they have been released. Behaviors on the other hand encapsulate complex functionality that are connected to certain interactions. Figure 2 illustrates the concept of EMIL behaviors. To employ behaviors, they can be added to widgets.

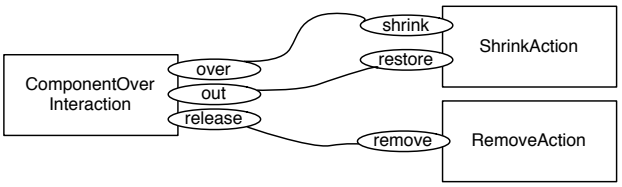


Fig. 2. The trash bin behavior is an example for an EMIL behavior. It basically shrinks a component that is dragged over the component that contains the trash bin behavior and restores the original size as soon as the component is dragged out. Shrinking and resizing gives a visual cue for the behavior's functionality. If the dragged component is released, it will be removed. The design of a behavior connects so called 'Interactions' with 'Actions': The interaction ElementOverInteraction has three outlets: 'over' which is fired whenever a component is dragged, 'out' whenever it is dragged out and 'release' whenever the dragged component is released above the component. These outlets are connected to inlets of actions. ShrinkAction shrinks and restores the size and RemoveAction removes the released component from the containing view.

The EMIL UI framework builds the foundation for the authoring application and programmers can enhance existing visual and non-visual components as well as create new ones.

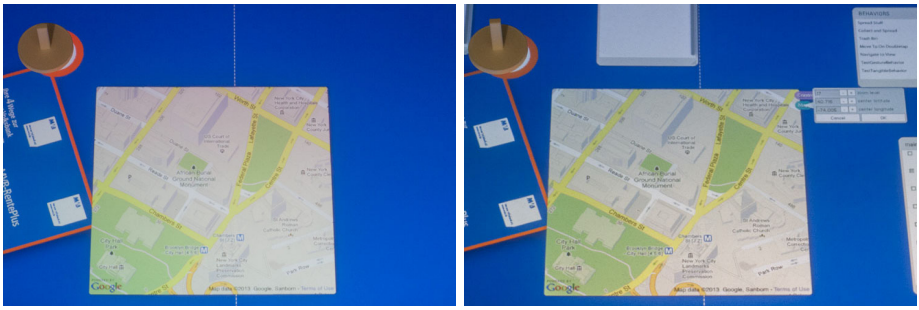
3.2 EMIL Authoring Application

The center of prototyping in the EMIL process is the EMIL authoring application (EAA). In EAA, prototypes can be created and modified without coding. EAA knows two modes: The live mode allows using the application (see figure 3(a)) and the authoring mode enables designers and users to modify the prototype (see figure 3(b)). The authoring mode can be started by putting the so-called authoring tangible object on the surface or by using a key combination on the hardware keyboard. As soon as a user removes the authoring tangible or enters the key combination again, the authoring results will be saved and the live mode will be re-entered.

To actually create prototypes, designers gather around the IS in a collaborative work setting. Figure 1 illustrates such a prototyping session. After starting EAA, designers load an existing prototype or create a new one. A new prototype opens with an empty view. In the course of the prototyping, designers add new views and widgets to a prototype and configure those views and widgets.

The authoring process is based on a building block principle. Visual and non-visual components can be dragged out of menus onto the surface. For instance, a designer may drag a map widget out of the widget library menu onto the surface and edit its built-in map behavior by tapping the appropriate behavior plug (see figure 3)). Dragging a behavior out of the behavior menu onto a widget adds it. A new plug appears visualizing that the behavior has been added. The behavior can be customized similar to the built-in behavior. Tapping the behavior plug opens its properties menu allowing for manipulation.

Each prototype is stored in its own folder in a cloud-based storage (currently in the Dropbox[1]). Technically spoken, a prototype consists of resources stored in a file and folder structure. Adding resources like media files (pictures, photoshop files, video, audio) to the prototype's media folder makes them available in EAA. As every involved designer in the prototyping process can be invited to share the prototype folder in the cloud storage, they can create, modify or delete resources in the folder from every device connected to the cloud storage.



(a) (b)

Fig. 3. (a) A cutout of an EMIL prototype in the live mode in which two documents connected to tangible objects and a map widget are visible. (b) The same cutout as in (a) but in the authoring mode. On the right of the widgets, plugs allow accessing the widgets' properties menu. For instance, the map widget shows the menu of its built-in map functionality. In the authoring mode, additional menus appear on the surface that allow to add widgets, behaviors and views to the prototype.

Out of media resources stored in the cloud storage, designers can build their own widgets combining them with EMIL behaviors. For instance, dragging the image of a trash bin out of the Dropbox menu and subsequently adding a trash bin behavior to it creates a fully functional trash bin widget. Therefore, designers can prepare media resources in advance and use them in EAA. As Photoshop[2] is a popular application amongst UI designers, EMIL supports importing Photoshop files. Hence, designers can use their accustomed tools to create resources for

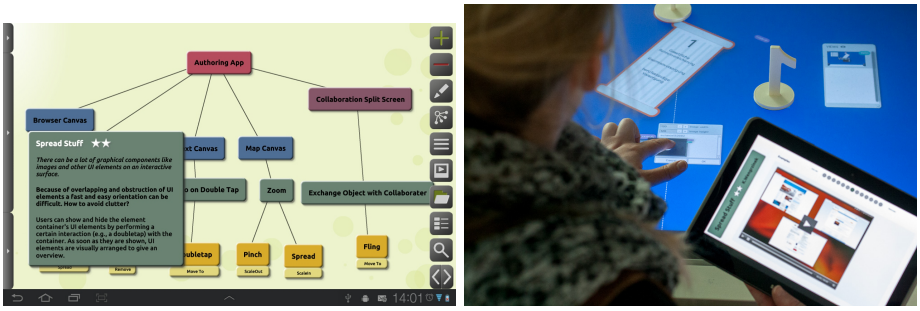
[1] `https://www.dropbox.com`
[2] `http://www.adobe.com/products/photoshop.html`

EMIL. During and after the prototyping session involved authors can iteratively refine the resources in the cloud storage.

3.3 EMIL Pattern Authoring and Browsing System

We introduce the EMIL Pattern Authoring and Browsing System (EPABS) that enables systematically building up and retrieving IS design knowledge. EPABS constitutes a database that stores information about the visual and non-visual components of the EMIL UI framework and prototypes created with EAA. We prepare this information in EPABS in the form of interaction patterns [1] and hierarchically arrange them to form an IS interaction pattern language similar to [13]. In the sense of [1], each EPABS pattern describes amongst others the problem that lead to its creation, its solution and examples for its usage in existing prototypes.

EMIL authors can browse and extend EPABS by using a tablet computer app (see figure 4(b)). In this app the pattern language is visualized by a node link graph (see figure 4(a)). Authors can apply visual filtering algorithms to the graph in order to narrow down the search for relevant patterns. After selecting a pattern, authors can read the pattern information or watch example videos or photos of its application in existing prototypes. If authors reuse an EPABS pattern in their own prototype they can attach videos and photos of its usage to the pattern's example section. Such videos and photos can be created employing the tablet computer app. Using the app, authors also add new patterns to the pattern language.



(a) (b)

Fig. 4. (a) Screenshot of the EPABS app on a tablet computer. It shows EPABS' pattern language represented by an interactive node link graph. In the graph, the 'SpreadStuff Behavior' has been selected showing its short description. (b) A designer add a SpreadStuff behavior to an image canvas after watching an example video of its usage in the EPABS app on the tablet computer.

4 Evaluation

We presented EMIL to a design team to gather qualitative feedback. This team consisted of a UI designer, a programmer and a concept designer from an advertising agency. This team had so far created four interactive surface applications. After the presentation, we interviewed them.

In their IS design process, the team usually creates prototypes for first tests that a programmer codes from scratch. This approach has at least three disadvantages. Firstly, it takes too long to create such a prototype. Secondly, the designers have to communicate their ideas to a programmer that has to convert them into code. Thirdly, they discover erroneous design decisions too late as the testing on the actual target platform comes too late in the process. With EMIL, they can quickly create prototypes themselves using the authoring tool without the need to communicate their ideas to a programmer and without writing code. This allows designers to "get their noses out of photoshop" and create prototypes themselves at an early design stage. In their opinion, EMIL provides a set of standard components for multi-touch and tangible user interfaces. Such components would already exist for mobile applications but not for IS. Such a set in combination with the authoring tool allows for the rapid prototyping of usable IS software. Instead of coding the prototype, the programmer involved in the design process can enhance EMIL's set of components and behaviors if necessary.

They embraced the iterative approach that allows alternately creating a prototype at the surface and preparing resources with their desktop tools allowing for quick design – test cycles. This offers to instantly see design results on the surface of resources created with accustomed tools like Photoshop giving a quick feedback to design decisions. They especially liked that authors involved in the design process meet in front of the IS to collaboratively assemble and modify the results of their work directly at the surface. Additionally, the combination of provided complex widgets and simple widgets they can build themselves combining graphical resources with EMIL behaviors makes sense to them, as it combines standard with custom functionality. However, they additionally desired a tool that allows building complex widgets without the need for programming.

The design team considered multiple uses of an EMIL prototype. Initially, it could be used to prepare a prototype in advance to a client meeting. Therefore, they could present this prototype to the client and potentially acquire a new job. Also, throughout the design process the prototype can be used to gather feedback with the client. If the prototype has matured, they can evaluate the prototype in a user test. Lastly, as the agency usually develops their IS software also in Flash, they can use the prototype's code and resources as a foundation for the final product development.

They suggested to provide EMIL to the open source community. Building blocks like behaviors or widgets could be easily extended by other Flash developers. This, however, led to their main criticism. They assume that there are currently too few building blocks like behaviors, widgets and application templates available in the EMIL UI framework. Additionally, they deem it necessary

to try out EMIL in a real project. Pertaining the UI of the authoring tool, they demand fewer windows for a better visual overview.

5 Conclusion and Future Work

The EMIL environment presents a rapid prototyping authoring approach for the creation of interactive surface software in design teams. It comprises UI and interaction components, an authoring tool and a database that provides storing and retrieving design knowledge using a tablet computer app.

Authors create the IS prototypes directly at the surface employing the EMIL authoring application (EAA) without the need to possess programming skills. Therefore, created software is tested early and frequently on the target hardware. Using EAA, designers configure components and their behavior. Additionally, simple components can be easily created based on resources that authors prepare and store in a cloud-based storage system.

In an expert interview we gathered qualitative feedback from a design team. The design team especially liked the iterative approach provided by EMIL and the possibility to create prototypes without programming knowledge using a 'standard' set of components. The team sees the potential to facilitate and speed up their own IS software design process employing EMIL.

In future work, we need to evaluate EMIL in a real design project. However, as the evaluation has shown, there are still too few components in the UI framework. Therefore, we need to enhance our set of components.

Acknowledgements. This research work has been financially supported by the BMBF-FHProfUnt grant no. 17043X10.

References

1. Borchers, J.O.: A Pattern Approach to Interaction Design. In: Proceedings of the 3rd Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques, DIS 2000, pp. 369–378. ACM, New York (2000)
2. Echtler, F., Klinker, G.: A Multitouch Software Architecture. In: NordiCHI 2008: Proceedings of the 5th Nordic Conference on Human-Computer Interaction, pp. 463–466. ACM, New York (2008)
3. Hansen, T.E., Hourcade, J.P., Virbel, M., Patali, S., Serra, T.: PyMT: a Post WIMP Multi-Touch User Interface Toolkit. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, ITS 2009, pp. 17–24. ACM, New York (2009)
4. Kaltenbrunner, M.: reacTIVision and TUIO: a Tangible Tabletop Toolkit. In: ITS 2009: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, pp. 9–16. ACM, New York (2009)
5. Khandkar, S.H., Sohan, S.M., Sillito, J., Maurer, F.: Tool Support for Testing Complex Multi-Touch Gestures. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (2010)

6. Klemmer, S.R., Li, J., Lin, J., Landay, J.A.: Papier-Mache: Toolkit Support for Tangible Input. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2004, pp. 399–406. ACM, New York (2004)
7. König, W.A., Rädle, R., Reiterer, H.: Squidy: A Zoomable Design Environment for Natural User Interfaces. In: CHI EA 2009: Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems, pp. 4561–4566. ACM, New York (2009)
8. Landay, J.A., Myers, B.A.: Sketching Interfaces: Toward More Human Interface Design. Computer 34(3), 56–64 (2001)
9. Laufs, U., Ruff, C., Zibuschka, J.: MT4j - A Cross-platform Multi-touch Development Framework. In: ACM EICS 2010, Workshop: Engineering Patterns for Multi-Touch Interfaces, pp. 52–57 (2010)
10. Lawson, J.-Y.L., Al-Akkad, A.-A., Vanderdonckt, J., Macq, B.: An Open Source Workbench for Prototyping Multimodal Interactions Based on Off-The-Shelf Heterogeneous Components. In: Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS 2009, pp. 245–254. ACM, New York (2009)
11. Luderschmidt, J., Bauer, I., Haubner, N., Lehmann, S., Dörner, R., Schwanecke, U.: TUIO AS3: A Multi-Touch and Tangible User Interface Rapid Prototyping Toolkit for Tabletop Interaction. In: Dörner, R., Krömker, D. (eds.) Self Integrating Systems for Better Living Environments: First Workshop, Sensyble 2010, pp. 21–28. Shaker Aachen (November 2010)
12. MacIntyre, B., Gandy, M., Dow, S., Bolter, J.D.: DART: A Toolkit for Rapid Design Exploration of Augmented Reality Experiences. In: Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology, UIST 2004, pp. 197–206. ACM, New York (2004)
13. Remy, C., Weiss, M., Ziefle, M., Borchers, J.: A Pattern Language for Interactive Tabletops in Collaborative Workspaces. In: Proceedings of the 15th European Conference on Pattern Languages of Programs, EuroPLoP 2010, pp. 9:1–9:48. ACM, New York (2010)