

Towards an Integrated Process for Interactive Surface Application Development

Tobias Hesselmann

OFFIS Institute for IT
Escherweg 2
26121 Oldenburg, Germany
tobias.hesselmann@offis.de

Susanne Boll

University of Oldenburg
Escherweg 2
26121 Oldenburg, Germany
susanne.boll@uni-oldenburg.de

Wilko Heuten

OFFIS Institute for IT
Escherweg 2
26121 Oldenburg, Germany
wilko.heuten@offis.de

ABSTRACT

The specific characteristics of ITS systems introduce many challenges and questions regarding application development for such systems. Most notably, developers are faced with the question what methods they should employ to build efficiently and effectively usable applications for ITS devices. On the one hand, abstract and device agnostic design processes, such as the human-centred design process, provide a basic outline for development. On the other hand, a multitude of specific methods, such as device specific guidelines or design patterns for ITS systems, provide very concrete guidance on a finer granularity level. In this paper we argue that both extremes are not sufficient to guide developers throughout the development process alone, and that there is demand for an integrated development process, which specifically considers the specifics of ITS systems. With SCIVA, we propose such a process, which focuses on the gestural and visual affordances of ITS systems.

Keywords: Process Models, Interactive Tabletops and Surfaces, Surface Computers, Multi-touch

INTRODUCTION

Developing applications for Interactive Tabletops and Surfaces (ITS) is often a demanding task for engineers, particularly if they are new to the field of ITS computing. Typically, they are used to develop applications for Desktop-PCs, basing their designs on the established WIMP (Windows, Icons, Menus, Pointing Device) paradigm for interface development.

Unfortunately, the form, size, interaction concepts and affordances of ITS systems are inherently different from Desktop PCs, introducing many challenges for application development on such systems (also see [1]). Ignoring the specific characteristics of ITS systems, i.e. applying the same concepts known from Desktop PC development without adaptation, can negatively impact the user experience in final applications. Thus, developers need clear guidance when creating application for ITS systems.

Nevertheless, research in this field is still rather scarce. On the one hand, one can find abstract UI design processes, such as the well-known Human Centered design process (HCD) [1]. HCD structures UI development in four distinct phases: A context of use analysis, a requirements

definition, the actual design phase, and an evaluation of the resulting interface. The process is then iterated until a sufficient result was reached in the evaluation.

While the process adds value in providing a basic outline for user interface development, it is device-agnostic and consequently does not make clear statements about the concrete organization of the design phase and about appropriate design methods for the particular field of interactive tabletops and surfaces, leaving ITS developers without concrete guidance at this point.

On the other hand, we can find fairly concrete and device specific guidelines on varying levels of granularity. For example, research has developed user-centric methods for the definition of gestures for interactive tabletops [3], or proposed the use of design patterns for ITS systems [4]. In addition, we can find sets of guidelines and best practices compiled by the industry, e.g. user interface guidelines for the Microsoft Surface [5]. The available guidelines concentrate on fairly different levels in ITS development, again leaving developers without clear guidance when and where to apply the appropriate methods. Moreover, the question arises whether these guidelines and methods cover essential aspects that ensure a high usability of ITS applications.

Between these two extremes, we can currently not observe an integrated design process that (1) decently structures the development process so that it can easily be followed by UI engineers and (2) at the same time provides concrete guidance considering the essential characteristics of ITS.

On EICS 2011, we propose such a process – SCIVA (Surface Computing for Interactive Visual Applications) [6]. In the following, we shortly describe the outline of the process, while we refer the reader to [6] for a more detailed description of the process and its individual steps.

THE SCIVA DESIGN PROCESS

To approach the issues described before, we created an iterative design process called SCIVA [ʃi:va], which aims to structure the development process and consider essential aspects in ITS development, while remaining flexible enough to apply different methods depending on the given task and scope of the application.

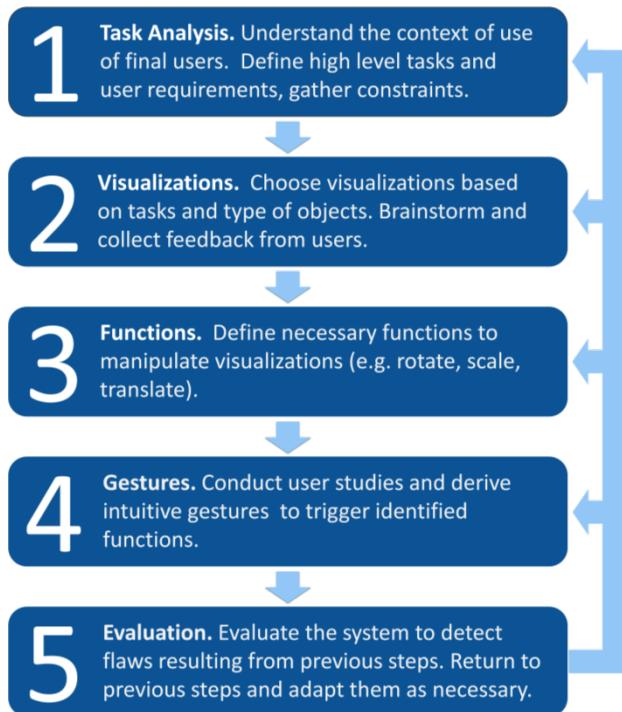


Figure 1: Outline of the SCIVA design process

SCIVA is an iterative process, which consists of five distinct steps (see Figure 1). The structure and order of the different steps follows a workflow that is derived from experiences obtained in the development of ITS applications at our lab, as well as research of related work. Each step can be instantiated with different methods depending on the task given. As a guideline, we propose user-centric methods based on the current state of the art in [6]. The process is iterated after the fifth step (evaluation) at latest, but iterations are also possible in the individual steps, as necessary. Of course, it is also possible to skip individual steps if necessary and meaningful.

Task Analysis

SCIVA starts with a *Task Analysis*, which frames the development of the application. Typically, this step includes a context of use analysis and the definition of requirements for the application, corresponding to the first two steps of the HCD process. In addition, we propose to define one or more high level goals for the application to sharpen the development process and ensure that all development efforts are aimed towards that particular goal. This approach is inspired by the definition of a development “metaphor” known from extreme programming, which pursues a similar goal. Additionally, we propose to define development constraints, such as particular software APIs that need to be used, or particular hardware platforms to develop against. After this step, the developer should have a clear picture of the available space of development possibilities. The following three steps, i.e. the definition of visualizations, functions, and gestures, can be seen as an instantiation of the design phase of the HCD process, structuring the phase for ITS development.

Visualizations

As the second step, we propose to define appropriate *visualizations* for the application. This step is particularly important, as visualization and interaction are typically closely intertwined in surface computing. Mostly, manipulations of visualized objects will be carried out by directly touching those visualizations with the fingers or hands. Thus, this aspect should be decently considered in the design of appropriate visualizations. As a starting point, we propose to select suitable representation for the data that will be visualized in the final system, then brainstorm and gather feedback with final users, and finally optimize the resulting visualizations to support the characteristics of ITS systems. Critical aspects to consider at this point will include occlusions by the users extremities, varying positions and orientations of the user, and the distance between user and visualized content. Again, we refer the reader to [6] for a more detailed description of these issues.

Functions

In the third step *functions* are defined to manipulate the visualizations defined before. The outcome of this step answers the question: Which functions are needed to manipulate the visualizations of the application, and how should its look change when applying a particular function to it? This typically includes atomic operations (see [7]), such as moving, scaling, or selecting an object, as well as more sophisticated and application specific functions.

It is important to note that these functions should remain independent from gestures or other interactions carried out by the user, i.e. they could be triggered by the system itself without any interaction of the user. The idea behind the decoupling of functions from the actual user interactions is that developers can concentrate on the functionality of the system first, and changes on the interaction level of the system will be more lightweight in later iterations.

Gestures

The fourth step is dedicated to the definition of appropriate *gestures* to trigger the functions defined in step three. While other methods of interaction apart from touch based interaction might be feasible in certain situations, such as the use of haptic devices (tangibles), touch-based and gestural interaction is by far the most established and common interaction method for ITS, which made us focus this aspect in the process. Several methods for gesture definition have been proposed, including predefined sets of gestures, such as the Open Exhibits Gesture Library [8], or user-centric methods, such as the well-known approach of Wobbrock et al. [3]. According to our experiences, the method of choice should be adapted depending on the tasks, the domain and the environment of the system.

An important aspect at this point, which is often not considered appropriately, is the tradeoff between efficiency and intuitiveness of a gesture set. Most approaches for

gesture definition primarily target the immediate usability (or intuitiveness) of a gesture set, while in long-term usage, the efficiency and effectiveness of a gesture set plays a much more important role. For example, aspects such as the physical strain imposed by a gesture, as well as the time needed to execute a particular gesture do not play an important role for first time and occasional users, but might heavily impact the user experience for expert and frequent users. We thus recommend to carefully ponder these aspects when defining a gesture set for an ITS system.

Evaluation

An iteration of the process is concluded by an evaluation of the system as a whole. This step can be considered as analogous to the evaluation phase in the HCD process. Primarily, it should provide insights regarding errors and design flaws in the application that serve as foundation for follow-up iterations of the process. Typically, user-centric methods, such as think-aloud evaluations will be employed in this step. Depending on the outcome of the evaluation, application designers and final users will decide whether follow-up iterations are necessary, and which steps of the process will need to be repeated or adapted. For example, if the chosen visualizations turn out insufficient, the second step is repeated, and all follow-up steps are also checked for necessary changes.

CONCLUSION AND OUTLOOK

In this paper, we argued that development of applications for ITS bears many challenges, particularly for developers new to the field of surface computing. Existing processes and methods are either device agnostic, such as the established HCD process, and consequently do not consider the specific characteristics of ITS systems. Or they are fairly concrete, such as design patterns for ITS, but do not structure the design process. As a solution, we have proposed an integrated UI design process tailored to ITS development, which developers can hold on to when designing for ITS systems.

SCIVA is derived from several years of ITS development experience in our lab, as well as thorough research of related work related to existing challenges in ITS development. Based on our experience, the process provides clear indications and is particularly helpful to developers new to the field of ITS development. To validate and refine the process, we are currently applying SCIVA in three ongoing projects involving teams of 1-10 developers, collecting feedback and carefully observing drawbacks and advantages in the development process.

BIOGRAPHY OF THE AUTHOR

Tobias Hesselmann graduated from Oldenburg University, Germany, with a Diplom degree in 2007. Since then, he is PhD student and associate researcher in the Intelligent User Interface group at the OFFIS Institute for Information Technology, Oldenburg, Germany. His main research targets ubiquitous systems, with a strong focus on interactive tabletops and surfaces.

Tobias is actively researching the field of interactive tabletops and surfaces since 2007. His work includes tabletop-based Information Visualization and Visual Analytics Systems (e.g., [9, 10]), methods and design aspects related to the usability and user experience of ITS systems (e.g., [6], and interactions between ITS devices and other ubiquitous systems, such as mobile phones (e.g., [11]).

REFERENCES

1. Bachl, S., Tomitsch, M., Wimmer, C. Grechenig, T. Challenges for Designing the User Experience of Multi-touch interfaces. Workshop on Engineering Patterns for Multi-Touch Interfaces, 2010.
2. International Organization for Standardization. ISO 9241-210: Human-centred design for interactive systems. <http://www.iso.org>, 2010.
3. Wobbrock, J. O., Ringel Morris, M., Wilson, A. D. User-Defined Gestures for Surface Computing. In *Proc. CHI 2009*, pp. 1083-1092.
4. Christian Remy, Malte Weiss, Martina Ziefle and Jan Borchers. A Pattern Language for Interactive Tabletops in Collaborative Workspaces. In *Proc. EuroPLoP 2010*.
5. Microsoft Corporation. Microsoft Surface User Experience Guidelines. Available on MSDNAA. 2009.
6. Hesselmann, T., Boll, S. Heuten, W. SCIVA - Designing Applications for Surface Computers. To appear in *Proc. EICS 2011*. ACM, 2011.
7. Aliakseyeu, D., Subramanian, S., Alexander, J. Supporting Atomic User Actions on the Table. In *Tabletops – Horizontal Interactive Displays*. Christian Müller-Tomfelde (ed.). Springer, 2010.
8. GestureWorks. Open Source Gesture Library. <http://openexhibits.org/gesturelibrary>. Date of access: 2011/02/07
9. S. Flöring, T. Hesselmann. TaP: Towards Visual Analytics on Interactive Surfaces. In *Proc. COVIS 2009*, Technical Report, 2010.
10. T. Hesselmann, S. Flöring and M. Schmitt. Stacked Half-Pie Menus – Navigating Nested Menus on Interactive Tabletops. In *Proc. ITS 2009*. ACM, 2009.
11. T. Hesselmann, N. Henze, S. Boll. FlashLight: Optical Communication between Mobile Phones and Interactive Tabletops. In *Proc. ITS 2010*. ACM, 2010.