

---

# Interaction Toolkits

## **Paul Holleis**

University of Munich (LMU)  
LFE Medieninformatik  
Amalienstr. 17  
Room 206, 2<sup>nd</sup> floor  
+49 89 2180 4657  
paul@hcilab.org  
<http://www.hcilab.org/paul>

PhD-Supervision: **Dr. Albrecht Schmidt**



## **Abstract**

Ubiquitous computing is no longer the playground of hardware specialists and programming geeks only. Platforms and components for hard- and software are readily available to create complex applications. However, this often requires detailed knowledge in several areas like electrical engineering, protocols, programming, etc. Therefore the design process should be supported by tools. In my thesis I try to spot the parts of such processes that can be simplified, verified or automated. This includes the choice of interaction, the choice of hard- and software, deployment, and the evaluation of the resulting system. I also anticipate providing tools that can validate certain properties and show key characteristics of developed applications.

## **Keywords**

Ubiquitous and pervasive computing, interactions, tool support, automation, verification

## **Problem Statement and Research Question**

Researchers in the Ubicomp community are often working with a variety of technologies and techniques. This includes sophisticated hardware devices such as PCs, Laptops, PDAs, mobile phones, etc., as well as small microcontrollers and single sensors. Applications range from simple, contained and centralised to complex, distributed and dynamic systems. Many aspects like privacy, security, complexity, robustness,

user acceptance and user friendliness etc., play important roles in the design of such applications. Such design processes require the commitment of many different types of professions: Market analysts, user interface designers, programmers, mechanical and electrical engineers, to name but a few. Often whole projects are done by people with significant knowledge in only one or two of those areas. All other aspects are neglected, done by inexperienced people or outsourced.

The aim of this work is to provide a system that incorporates knowledge and experiences from a broad range of developers and research to enable others to profit from that and to simplify the development as well as evaluation of applications. This research is supposed to help application developers from various backgrounds to implement their ideas quickly and comfortably, supporting them to avoid common errors and improve efficiency, usability, stability, etc. using expertise from other projects.

This includes support of approaches that have been found very important in standard software engineering. Applications have many interesting characteristics like the longest or average path to navigate a menu to some specific item or the number of different possibilities to initiate a certain action. Knowing such values helps creating better and more robust designs and can often be retrieved automatically. Another significant aspect is the validation of defined properties. It is, for example, in many applications important to ensure that it is possible to navigate back to a main menu from every possible state. Some application behaviour can also be automatically inferred. In [5], for instance, Harold Thimbleby shows an example where matrix algebra can be used to check what happens

when a certain button is pressed twice (regardless of the current state of the system) although the designer has only specified the behaviour after one single press.

One of the central questions that I look forward to answering is where exactly people need toolkit support and what features such support will have to provide and what issues it would have to avoid. This implies looking at how to incorporate knowledge and experiences that result from both theoretical and (often not formalised) experimental experiments.

### **Approach and Methodology**

This thesis is done in the context of the research group Embedded Interaction [1]. In the past years, my colleagues and I have conducted several projects using different hardware platforms and a multitude of sensors and actuators. Applications range from self-contained, autonomous learning applications to wirelessly connected ambient displays. This helped to get insight into how such applications are realised and potential problems in the design and implementation phase.

These experiences combined with a careful research of experiences and work done by other researchers in the last years as well as ongoing and planned projects will serve as a basis to extract enough knowledge to be able to understand the process and specify requirements. The design processes and tools found will be compared with those of known software engineering and different steps and tasks performed during that process will be discovered. The next step will then be to identify possible spots for integrating tool support.

To be able to evaluate the usefulness of such tools, each identified possibility will be rated and

prototypically implemented. I envision applying those to our own projects as well as letting different user groups try those systems. To get results, some these groups will first try to accomplish a set task without the tool and then with help of the tool. The other half will do the same, only vice versa. This helps to reduce the influence of implicit improvement when repeating a similar task. Experiences will be recorded and differences measured. According to the results there will be one or more iterations on the system.

It is expected to combine theoretical and experimental research using theoretical and experimental results to create an interesting mix of experiences and formal / statistically derived results.

### **Related Work**

There have been and still emerge several projects that develop frameworks and toolkits for application development with different technologies. This includes efforts to combine several programming languages (like in Visual Studio .NET) and general description languages of sensors, actuators and their data (like SensorML [6]). There are also several projects that brought forth toolkits for the implementation of ubiquitous applications:

The ECT Toolkit [2] was developed as part of the Equator project. It features a components and plug-in based architecture. Instances of components are used as nodes of a graph. The interface of each component defines outgoing (events) and incoming edges (operations). Application logic can be implemented by connecting outputs of components with inputs of others. A problem is that the huge number of available (and necessary) components renders the toolkit hard to

use. To implement more complex behaviour, complex components offering scripting capabilities are needed reducing the ease of use of the concept. This system assists in quickly prototyping applications.

Our own first steps towards toolkit support build on ideas from the iStuff toolkit [3]. It offers a variety of devices acting as input or output instruments. Each of those communicates with a proxy running on a computer. This computer is connected to a tuple space implementation. All messages sent to / from device proxies or applications are passed through this tuple space. An interesting feature of the toolkit is that an intermediary allows for dynamically remapping events, i.e. events can be redirected to another device during run-time. That system adds, among other things, toolkit support for dynamically exchanging components.

The Stanford HCI Group has begun to use iStuff. Lately they have developed d.tools, an architecture with toolkit support for rapid prototyping. Hardware is assembled using physical controllers, sensors and output devices [4]. The behaviour of the system can be implemented using a graphical editor implemented as a plug-in for Eclipse Development Environment. Users graphically arrange iconic representations of hardware items into a state graph where states specify device outputs and state transitions are triggered by physical inputs. The state graph can become quite complex and confusing even for a small number of buttons etc. It supports constant synchronisation of the software with the available hardware and its states.

Besides support for connecting different components these toolkits do not offer additional assistance with respect to verification, usability, security or efficiency.

## **Preliminary Results**

In our projects, many students have used different hardware platforms and lots of different types of sensors and actuators with various access protocols and communication abilities. I have observed that the first steps in using existing and new devices to create new applications are often quite hard. This can partly be simplified by the toolkits described above. However, I also found that from a higher level point of view, there arise many problems that cannot be solved in this way. Decisions of which interaction technique, which hardware and what values for parameters (like sensor rate, size) might best be used are still to be made by the application designer. Application logic verification and characteristics extraction is not used at all. Although it will not be possible to solve all of these issues automatically, I see great potential to add tool support for many of those aspects.

I am currently in the process of developing a toolkit for interaction applications [7]. It started to provide functionality similar to that of those projects described in the related work. However, I want to incorporate different views and interactions on the same structure to enable different types of users (programmers, designers, prototype implementers) an equally comfortable tool. In parallel, I endeavour to collect necessary information and to design tool support for other stages of design and higher level issues that I identified.

## **Conclusions and Future Steps**

To summarise, I want to define spaces where application development including external hardware devices can be supported with tools. At least prototypical implementations of such tools and their

experimental evaluation are planned. This requires gathering and categorizing of users, experiences, technology, design processes, etc. and the application of software engineering processes like validation and evaluation techniques.

I strongly depend on the collection of experiences and knowledge of other researchers and research groups as well as potentially industrial design work. This will provide me with the necessary details of where problems in such processes arise and where and in what extent toolkit support can be provided.

## **References**

- [1] Research Group Embedded Interaction Web Page <http://www.hcilab.org>
- [2] C. Greenhalgh, H. J. Izadi S., Mathrick J., and I. Taylor. ECT: A Toolkit to Support Rapid Construction of Ubicomp Environments. In Conference on Ubiquitous Computing (Workshop on System Support for Ubiquitous Computing UbiSys04), 2004
- [3] R. Ballagas, M. Ringel, M. Stone, and J. Borchers. iStuff: A Physical User Interface Toolkit for Ubiquitous Computing Environments. In Proceedings of the ACM CHI 2003 Conference on Human Factors in Computing Systems, pages 537-544, Ft. Lauderdale, USA, 2003
- [4] B. Hartmann, S.R. Klemmer, and M. Bernstein. d.tools: Integrated Prototyping for Physical Interaction Design. Stanford University Computer Science Technical Report # CSTR2005-06, 2005
- [5] H. Thimbleby. User Interface Design with Matrix Algebra. ACM Transactions on Computer-Human Interaction 11(2): 181-236, 2004
- [6] The SensorML Web Page <http://vast.nsstc.uah.edu/SensorML/>
- [7] EIToolkit: Toolkit for Embedded Interaction <http://www.eitoolkit.de>