Bettina Conradi, Doris Hausen, Fabian Hennecke,
Max-Emanuel Maurer, Hendrik Richter,
Alexander Wiethoff, Heinrich Hussmann (Editors)

Prototyping

Hauptseminar Medieninformatik WS 2009/2010

Technical Report
LMU-MI-2010-1, Feb. 2010
ISSN 1862-5207

University of Munich
Department of Computer Science
Media Informatics Group

Bettina Conradi, Doris Hausen, Fabian Hennecke, Max-Emanuel Maurer,
Hendrik Richter, Alexander Wiethoff, Heinrich Hussmann (Editors)

Prototyping

An overview of current trends, developments, and research
in Prototyping

Preface

This report provides an overview of current applications and research trends in the field of prototyping. There are application domains where prototyping plays a vital role. They range from traditional graphical user interfaces (GUIs) to ubiquitous computing applications, like tangible or haptic user interfaces and interactive surfaces.

During the winter term 2009/2010, students from the Computer Science Department at the Ludwig-Maximilians-University in Munich did research on specific topics related to prototyping and analyzed various publications. This report comprises a selection of papers that resulted from the seminar.

Each chapter presents a survey of current trends, developments, and research with regard to a specific topic. Although the students' background is computer science, their work includes interdisciplinary viewpoints such as theories, methods, and findings from interaction design, ergonomics, hardware design and many more. Therefore, the report is targeted at anyone who is interested in the various facets of prototyping.

In addition to this report, there are slides from the students' talks available at http://www.medien.ifi.lmu.de/lehre/ws0910/hs/ .

Munich, February 2010

The Editors

Bettina Conradi, Doris Hausen, Fabian Hennecke, Max-Emanuel Maurer,
Hendrik Richter, Alexander Wiethoff, Heinrich Hussmann

Contents

From Paper Prototyping to Sketching with Hardware

Eduard Held

Abstract— In this paper, two prototyping techniques, paper prototyping and experience prototyping, are described and compared to each other in order to give a potential lookout of future prototyping. Paper prototypes are mostly hand-drawn prototypes of an user interface which enable quick usability testing and improving with users before the actual programming starts. In experience prototyping, developers, users and clients actively collect experiences with products that have to be developed or changed, by interacting with prototypes. These experiences help understanding existing situations, exploring ideas and communicating the product vision. While paper prototypes have been used for decades for pure screen based human-computer interaction, new technologies allow more hybrid ways of interaction, which demand for more versatile usability engineering methods, like experience prototyping.

Index Terms—prototyping, paper prototyping, experience prototyping, discount usability engineering, guerrilla hci

1 Introduction

Developers of user interfaces use prototyping, especially low-fidelity prototyping, to realize and evaluate basic design ideas in early stages of the development. At this point, where changes are made quickly and frequently, prototypes need to be cheap, easy to make and may be limited in function, since they are only used to evaluate specific components and communicate basic concepts [9].

One of the most popular low-fidelity prototyping techniques is paper prototyping: Using only common office supplies like pencils, paper and scissors, user interface mockups can be created within hours. These paper prototypes can easily be adopted to the rapid changes and provide a very efficient tool for the classical user interface design [15]. But as technology evolves, new ways of human-computer interaction (hci) emerge and quickly move away from pure screen based interactions. To prototype those hybrid and often more complex interactions, developers require prototyping techniques that allow them to sketch with the hardware they are developing and provide tools for a more versatile product evaluation. One of those techniques is experience prototyping [1]: developers and users actively interact with prototypes and collect subjective experiences, which provide a better understanding of the product that has to be designed or improved.

The goal of this paper is to give a detailed overview of paper prototyping and experience prototyping, look at the pros and cons of each technique and compare them to each other. Based on the comparison I will also try to find out how prototyping will look like in the future. Another subsection will include an overview of guerilla hci [6], which describes discount usability engineering as a method to get developers started with prototyping in case they haven't been using it yet.

In section 2 detailed descriptions of paper prototyping, experience prototyping and guerilla hci are presented. Section 3 contains the discussion, which features the comparison and open questions. In the conclusions in section 4 I will then summarize the paper, include my own opinion on the topic and finally give a lookout of future prototyping.

2 From Paper Prototyping to Sketching with Hardware

This section includes individual descriptions and analysis of paper prototyping and experience prototyping based on the most important articles on this topic. Another subsection is guerrilla hci, which describes a technique to convince those developers, who haven't been using prototyping yet.

- *Eduard Held is studying Media Informatics at the University of Munich, Germany, E-mail: helde@cip.ifi.lmu.de*

- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009*

2.1 Paper Prototyping

In the development of user interfaces for human-computer interactions, one will always find some problems, which have to be solved. The earlier those problems are found, the easier they can be corrected. Paper prototyping *(see figure 2)* allows to find and eliminate basic problems with the help of end-users at a very early stage of the development cycle, often before any code has been written.

2.1.1 History

People started using paper prototypes in the mid 1980s [20], although not many developers have heard of its benefits at this time and it was only used by some of the pioneers of usability engineering. But when big companies, like IBM or Microsoft, started using paper prototyping in their product development process, paper prototyping started to become more popular [15]. Today, paper prototyping is a well known and accepted prototyping technique [16], [10], [15], [8].

2.1.2 Use

The typical paper prototyping process can be divided into three major steps: determining appropriate tasks, creating the paper prototype and usability testing.

In the first step, before actually creating the prototype, developers have to determine the tasks, which the users have to perform during testing. To support the improvement of the interface, these tasks have to come up to several requirements [15]: At first, the tasks should be realistic and pursue a goal which the user would also try to reach when using the final product in its actual environment. This helps simulating a real work situation and ensures that the test-user takes his/her job seriously. The next important requirement to a task is that the user should deal with the most critical components of the interface, those components which decide if the product is successful or not. The goal is to answer all questions of the developers regarding those parts of the product during testing. The scope of a task should be appropriate, meaning that the user has enough time to achieve a certain goal, but still provides enough feedback regarding the critical questions. It is also advised to keep the set of solutions for a certain task at a finite and predictable level, to limit the amount of work required for creating the paper prototype. Another recommendation is to create tasks with a clear end point, this ensures that the user, and no one else, decides, when a task is completed. This also contributes to a more realistic working situation. When creating tasks, it is also important to keep in mind, that most of the critical questions are answered by observing the user when he/she is interacting with the prototype, and not when he/she tells his/her opinion about the interface.

The next step is creating the prototype: Using common office supplies like paper, pencils, scissors and glue, a sketch of the interface can be created within a few hours. It is important to sketch each component onto a separate piece of paper, so that the components can be added and removed exactly as in the final product. Keeping the proto-

type colorless helps the user concentrating on the critical parts of the interface.

Opinions are divided when talking about what degree of detail should be implemented in the prototype. On the one hand, a highly detailed paper prototype contributes to a more realistic working scenario and prevents the user from guessing what to do according to the available buttons or menus. On the other hand, a too detailed prototype might distract the user from his actual task and lead him/her to focus too much on less important components. Lastly, since sketching on paper can be done by anyone, all people involved in the development of the product can and should take part in creating the prototype. This leads to a shared understanding of the product and allows everyone in the team to focus on the same issues. After the prototype is created, a final walkthrough is performed by the developers. This walkthrough, where the developers step into the role of users, is like a rehearsal of the usability test and helps finding and eliminating first problems.

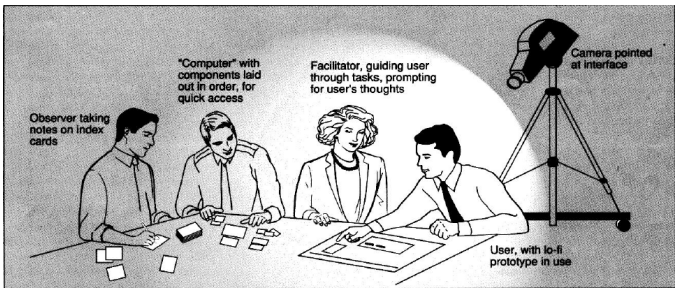


Fig. 1. A paper prototyping testing session. Image from Rettig [8].

In the third step the actual usability test is conducted *(see figure 1)*. There are several roles that have to be assigned to the team members, such as a computer, a facilitator and usually a few observers. The computer's task is to move the components of the paper prototype according to the actions of the user to simulate how the final product would behave when running on a real computer. The computer does not explain how the prototype works, but only reacts to the user's actions performed with the interface. The facilitator is the only person of the development team who may speak freely to the user during the usability test. According to Rettig [8], a facilitator has to perform the following three actions: giving the user instructions, motivating the user to express his/her thoughts during the test and ensuring that the test is done on time. This is an important role, because many things depend on the facilitators work, like how comfortable the user feels during testing, how well the user understands his/her task and how well the observers understand the user's problems. Hence the facilitator is usually someone who is trained in usability [14]. The observers are those people of the team who follow the actions of the user and make notes about the interaction of user and computer. They do not interfere in the testing process. Another important role in the usability testing is the user himself/herself, although this role is not filled by members of the development team. Test-users have to be chosen carefully to represent the set of users who will actually work with the final product. It is optional to use a camera pointed at the paper prototype to support later evaluation.

After testing, the team discusses the collected data and immediately changes the paper prototype accordingly. Since this process does not require any technical skills, every member of the development team, from programmer to management, can actively take part in changing the prototype. Testing and improving are performed until the development team has eliminated the most critical flaws and is ready to move on to the next step of development, which is redesigning the interface based on the collected data [15].

2.1.3 Advantages

The reason why paper prototyping is one of the most popular low-fidelity prototyping techniques today [16], [10], [15], [8] is that it has a lot of advantages to developers of user interfaces for human-computer interactions:

Using paper prototyping is an efficient way of including users into early product development: The resources needed for creating and changing paper prototypes are common office supplies like paper, pencils and scissors and every developer of user interfaces has access to those materials. The time required for creating, testing and changing a paper prototype is comparatively short [13]. Also, developers don't require any special skills to make use of paper prototyping, which makes it quite easy to use [8]. Although it is easy and fast, comparisons show that paper prototyping can disclose a similar number of critical problems as prototyping techniques with higher leves of fidelity [3], [18].

Paper prototyping allows developers to improve the usability of an user interface before programming it [14]: by sketching the interface on paper and testing it with end-users, real problems can be found very early. This way the programmers can be more confident that they are implementing the right product. This is also very helpful to validate, if the instructions from the internal marketing department are implemented correctly. Another benefit of finding problems before programming is that designers show a greater willingness to change a product they have put minimal time in rather than one they spent weeks developing [2]. And since testing and changing the paper prototype does not require much time, many tests are possible in a short period of time, which also contributes to improving the usability in this early stage of development.

Another advantage is that users are more likely to criticize the functionality rather than the design when working with paper prototypes [14]. It has also been found that users feel more comfortable when criticizing an interface that is still in development rather than a finished product [2].

When users are included from the beginning in the development of a product, the developers have a greater chance of finding out what the users really want and what they need. This increases the probability that the user will accept and use the product.

Paper prototyping can also be used as a tool to explore tasks[16] and to communicate the interface[10]: when developers try to find out how the users want the tasks to be like, paper prototyping can be used, instead of the more expensive field researches or interviews, in a way that developers and users create the tasks together by modifying the prototype. Those prototypes can then be used, instead of reading specs, to demonstrate to other members of the development team, how the interface should work.

Since paper prototypes don't require extra skills to be created and changed, the whole development team can take part in the paper prototyping process. This strengthens the team spirit and establishes a common understanding of the product. Everyone involved in the development process can actively contribute to improving the usability.

2.1.4 Disadvantages

The first major problem with paper prototypes is that, when compared to a high-fidelity prototype, the total number of flaws found during testing can be significant lower [5]: Although many of the critical problems can be found by using a paper prototype, some critical and many less critical properties of the interface can't usually be identified. Also, when compared to a high-fidelity prototype, a paper prototype leaves out many details which makes it more difficult to use it as a template for later programming. Many decisions, which were considered less important in the beginning, have to be made by the programmer himself/herself, although those decisions could later have a big impact on the acceptance of the product.

Since paper prototypes are only a very rough representation of the final product, questions regarding details have to be answered by later prototypes with a higher level of fidelity.

Another problem with paper prototyping is the authenticity of the interaction. Several members of the development team are required to simulate a working product and evaluate the usability testing session. When interacting with a paper prototype, the user might have problems with several people watching him/her and waiting for actions to

perform [11]. Another issue regarding the interaction are the unrealistic answering times of the system.

When actions of the user would mean only little changes on the computer screen, paper prototyping is not appropriate to test whether the user noticed those changes or not since in testing the user sees that the computer lays additional things onto the table. Simulating scrolling is another weakness of paper prototyping: folding the paper or using cutouts are more noticeable and can lead the user to try those functions only because they are implemented. The "feel" of scrolling in a paper prototype also differs greatly from scrolling in computer programs.

Also, when developing content-rich products, paper prototyping is not appropriate, because implementing all the content would mean too much work. One possibility is to replace the real content by fake content that can be created or printed quickly, but that reduces the authenticity of the interface[16], which forces the user to imagine the real product. This makes it harder for the user to concentrate on his/her original task and decreases the chance of finding critical flaws.

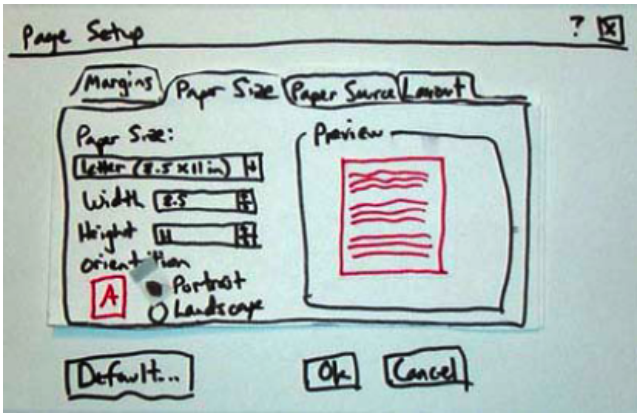


Fig. 2. Paper prototype. Image from Snyder [14].

2.2 Experience Prototyping

Paper prototyping is a very popular prototyping technique to improve the traditional, pure screen based interactions. But as technology evolves, more hybrid ways of interaction emerge and demand for new prototyping techniques. One of these techniques is experience prototyping[1]: to explore and understand, what future products will behave like, developers and users actively collect experiences by interacting with prototypes and sketching with the hardware they are developing.

2.2.1 History

Experience prototyping was introduced by Buchenau et al [1] in the year 2000: where prototyping techniques always deliver some kind of experience, the difference of experience prototyping is that it is all about gaining subjective experience by active interaction. While common prototyping methods are often used with a mainly passive audience [1], experience prototyping involves developers as well as clients and future users actively in the development process.

2.2.2 Use

Buchenau et al [1] have identified three different kinds of activities that can be supported by experience prototyping: (1) understanding existing user experiences and context, (2) exploring and evaluating design ideas and (3) communicating ideas to an audience.

When trying to understand existing user experiences, experience prototyping can be used to simulate the contextual factors that influence the use of the product and help finding problems and opportunities. Buchenau et al [1] provide three examples for this area of application that are summarized in the following in order to demonstrate the use of experience prototyping. The task in the first example was to create a product to support patients with chest-implanted automatic defibrillators. Since the developers did not have any experiences with this kind of situation, every team member was given a pager for a few days. To simulate a defibrillating shock, pager signals were sent at random times. The results were first hand experiences gained by the team members, who had a better understanding of the users' situation afterwards. This example shows that some final products don't allow the developers to use them (unless they get a chest implanted defibrillator), but experience prototyping still can provide a better understanding through subjective experiences. The second example was about a development team having to create an interface for an underwater remotely operated vehicle. By creating a situation similar to the one a pilot would be in, the team collected personal experiences, which enhanced the developers' understanding of the pilot's problems and provided a common reference point for further development and communication with real users. In the third example, a team had to find out the needs of passengers to create a new rail service. The methods used to get an understanding of those needs included role playing and bodystorming. Role playing is a technique where developers and users act like they were in a certain situation to get a shared understanding of problems, just like the prior examples show. Simsarian [12] confirms the advantages of role playing: (1) maintaining group focus, (2) creating a shared understanding, (3) building a deeper understanding grounded in context and (4) "the ability to viscerally explore [new] possibilities". Svanaes et al [17] also recommend using role playing and present a detailed workshop structure in their paper. Bodystorming is often a part of roleplaying and describes the process of brainstorming and discussing solutions in a real or similar environment of the final product [7]. Oulasvirta et al [7] described it as "best suitable for designing [...] activities that are accessible and unfamiliar to the researchers". The use of these two techniques allowed the developers of the third example to make discoveries of personal significance which helped understanding and discussing them in the team and with users. The personal experiences created lasting memories which helped the designers during further development.

Developers also use experience prototyping to explore and evaluate design ideas [1]: personal experiences simplify the discovery of appropriate solutions to a problem and also help evaluating them with team members, users and clients. Two examples help understanding the concept: The first example is about developing a control device for a video game, where the team discovered three different directions for their development. Simple objects representing the different user interfaces were used to gain a first understanding of the user experience. This way the designers could quickly disclose some basic problems and benefits of the different solutions without creating the real products. In the second example, a team had to come up with ideas for the interior layout and components for an airplane. Bodystorming was used in a full-scale foam-core environment, which simulated the interior of the plane. With the help of chairs and other office components, various social situations were simulated in different arrangements. The result was a better and common understanding of users' experiences, which was achieved quickly and at low cost.

Experience prototyping can also be used as a tool to communicate design ideas [1]: by experiencing it, team members, users and clients are more likely to properly understand an idea. Another two examples shall help the reader's understanding. The first example is about the development of an early device for digital photography *(see figure 3)*. After noticing that the clients did not fully understand the intended user experience and product behavior using traditional communication methods, which created passive experiences, they decided to create a "look and feel" prototype. The new prototype enabled the clients to actively interact with the product and understand the intentions of the developers. This was a crucial part in convincing the clients and led to the success of the product. The second example shows how passive communication methods, in this case video, and experience prototyping can collaborate: The "Kiss Communicator" is a product to exchange emotional content between people separated by physical distance. To communicate the concept, the developers decided to use a video, rather than a "look and feel" prototype, which showed how a

dreamy couple, who are working apart, used the device. This way a more appropriate atmosphere was created to communicate the product experience to the clients.

Fig. 3. Digital camera interaction architecture prototype. Image from Buchenau et al. [1].

2.2.3 Advantages

The first benefit from experience prototyping can be illustrated by quoting the chinese philosopher Lao Tse: "What I hear I forget. What I see, I remember. What I do, I understand!" By active interaction, subjective experiences are gained. One the one hand, those experiences help developers to understand users and the problems they have. One the other hand, users and clients get a better understanding of the developer's ideas [1].

Experience prototyping can simulate important parts of relationships between people, places and objects. Hence it is a good tool for developers to understand situations and products that they are unfamiliar with [1].

Developers can get inspiration, confirmation or rejection of ideas, according to the quality of experience they cause [1]: Users may like or dislike what a certain prototype looks and feels like. They can also inspire the developer by the way they interact with the prototype.

Using experience prototyping leads to a shared experience [9]. This does not only help the members of a development team to build a common vision, but also provides a shared point of view for developers, users and clients. This way, the work of a development team can be effectively lead into one direction. This direction can be determined collectively by developers, users and clients, which are able to speak in a "common language" [9] because of the shared experiences.

When using low-fidelity prototypes in experience prototyping, multiple prototypes can be created and evaluated in a short time and at a very early stage of the development [17]. This increases the chance of finding core problems and their solutions.

2.2.4 Disadvantages

The first problem is that we cannot actually be other people and can't experience what they experience [1]. Experiences are very subjective and have often been created over a long period of time. It is obvious, that no one can make the exact same experiences by simulating the environment for a few days. Still, experience prototyping can be helpful to provide a basic understanding of other people's situations with relatively small time effort.

Another problem is that experiences can not be predicted [1]. The creation of experiences involves many personal and contextual factors, where only the least ones can really be controlled. As a result, experience prototyping can be used to give a certain direction to the users' experiences, but one can't expect too much.

When conducting a roleplaying session, it could be problematic that everyone is actively involved. Active involvement means concentrating on personal experiences, rather than having an overview of the whole situation. Things, that affect the interaction of several people, like organizational components, might be overlooked. Involving one or more observers and a facilitator for each roleplaying group is highly recommended by Svanaes et al. [17].

Using experience prototyping as a tool to communicate design ideas can also have its problems. Developers have to carefully choose the degree of fidelity for a certain prototype: a level too low might require too much supervision to cause realistic experiences, whereas a high-fidelity prototype bears the chance that clients may become unshakably attached to early ideas as soon as they experience a convincing moment with the prototype [1]. To give a demonstration: the clients in the digital camera example, which is shown above, were so excited about the first hands-on experience with the prototype, that some details, which have been compromised by the designers due to lack of time, have simply been ignored and the design phase was announced as completed after the testing session, although nobody knew what impact those details had on the end-users' experiences.

As a conclusion, when applied with care, experience prototyping can be very helpful for designing interfaces for human-computer interactions.

2.3 Guerilla HCI

There are still developers, who don't have any experiences with prototyping [19] and are not using it in their development process. A possible explanation as well as a solution to this phenomenon is given by Nielsen [6]: By introducing "Guerilla HCI", he presents a method to facilitate the entry into usability engineering for software developers by using discount usability engineering.

Nielsen [6] calls the reason, why developers don't use usability engineering methods (not even the most basic ones like early focus on the user, empirical measurement or iterative design) "Intimidation Barrier". One big part of this barrier is that the costs of the recommended methods are perceived too high. The reason for this perception is that there are methods that are really expensive [4] and time consuming, but it is often forgotten to point out that there are cheaper methods as well. Another point is that the methods for testing the usability in hci are often intimidating in their complexity. Discount usability engineering addresses these two problems by providing methods that are (1) less expensive and time consuming and (2) less intimidating in their complexity, but still deliver valuable information for improving the usability of a product.

Discount usability engineering uses three techniques: (1) scenarios, (2) simplified thinking aloud and (3) heuristic evaluation. These techniques are not considered to be the best on the market, but rather are methods with comparatively low cost and complexity that still provide useful information. This makes them more likely to be used by usability engineering novices.

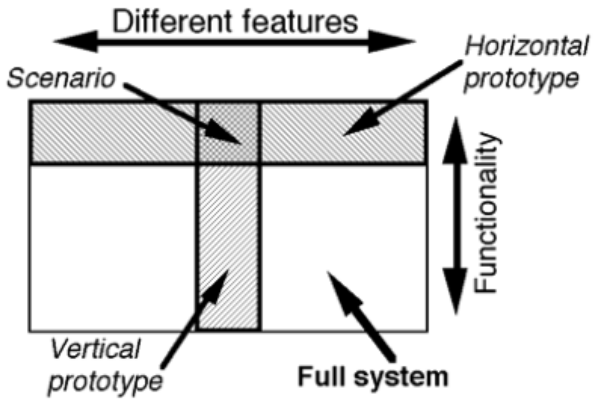


Fig. 4. "The concept of a scenario compared to vertical and horizontal prototypes as ways to make rapid prototyping simpler". Text and Image from Nielsen [6].

Scenarios *(see figure 4)* are a combination of horizontal and vertical prototypes [6]: Horizontal prototypes provide an overview of the functions of a product, but the single functions are not fully implemented and therefore not working. Vertical prototypes include only a few functions of the final product, but those are fully implemented. Scenarios combine these properties and result in a prototype that is drastically reduced in the total number of functions as well as in the depth of the specific functions. This provides a tool to cheaply, easily and quickly test a very specific part of the product to get fast and frequent feedback from users.

Simplified thinking aloud is a method where users are asked to think out loud while they perform specific tasks with a prototype. In contrast to the traditional think aloud method, no psychologist or user interface experts, who make detailed analysis using videotapes of the testing session, are involved. Another difference, which makes simplified thinking aloud more affordable, is that the number of test-users can be reduced, where a number of three to five test-users achieves the maximum benefit-cost ratio, according to Nielsen [6].

Heuristic evaluation is recommended instead of referring to current user interface standards and collections of usability guidelines, which have about one thousand rules to follow. By providing a small set of basic usability principles, developers can quickly get a first understanding. Yet, it is advised to acquire an outside usability expert to make sure that the principles are applied correctly and provide a maximum of valuable information [6].

Nielsen [6] conducted studies which confirmed that the discount usability engineering methods caused measurable improvements in usability and helped identifying the best design when several possibilities are given. Another study was made to determine the cost-benefit ratio of heuristic evaluation in a concrete example. Two things were required: (1) estimating the costs in terms of time spend performing the evaluation and (2) estimating the benefits in terms of increased usability. The result was a benefit/cost ratio of 48, which, although involving significant uncertainties, is a good reason for recommending the use of heuristic evaluation in the development process. The cost-benefit ratio of user testing was also estimated and showed a comparable value, hence user testing can also be of additional value for the designers.

3 Discussion

After seeing several reasons for using prototyping, there is still the question, which techniques is better for which situation. This section will present a comparison of paper prototyping and experience prototyping.

3.1 Paper Prototyping vs. Experience Prototyping

The main goal of paper prototyping, as well as of experience prototyping, is to improve the usability of an interface for human computer interactions. Although the methods used by the two techniques are largely different (testing a paper prototype vs. roleplaying and bodystorming), there are still some parallels noticeable: the first thing both techniques have in common is the early focus on the user. Paper prototyping involves the user to test basic ideas, sometimes even before a single line of code has been written. In experience prototyping, the users help understanding existing experiences and evaluating the designers' ideas early in the development process. Another parallel is the involvement of the whole team while using the techniques. Each member of the development team can actively participate in creating and changing a paper prototype. When conducting roleplaying- and bodystorming-sessions to explore ideas in experience prototyping, also the whole team is actively part of that process. In both cases, a "common language" [9] is provided, which helps concentrating on the same issues and therefore increases the chance of a better product. As described in [16], paper prototypes are used successfully as a kind of "working interview", instead of field research or interviews, to understand existing experiences. This is also one of the main activities that experience prototyping is used for. Both techniques also share the ability to operate as a tool to communicate ideas and concepts to team members, users and clients.

Besides the parallels, there are also some differences between paper prototyping and experience prototyping. The main one would be that paper prototyping is a single technique which is restricted to the medium "paper", whereas experience prototyping uses several techniques. Still, "Experience Prototyping is less a set of techniques, than it is an attitude, allowing the designer to think of the design problem in terms of designing an integrated experience, rather than one or more specific artifacts" [1]. This means, that experience prototyping primarily aims at creating a certain experience, where the different resulting artifacts are only a tool to provide this experience.

In paper prototyping, this final artifact is always a screen-based interface. As a result, experience prototyping may be able to create more and more versatile products and therefore experiences. The process of paper prototyping, with facilitators, observers and computers, also differs quite much from role playing, one of the core methods of experience prototyping: although both methods simulate certain conditions, paper prototyping focuses on the simulation the computer whereas roleplaying, partially conducted in the real environment, focuses on simulating experiences.

Considering the above, experience prototyping might be the more appropriate prototyping technique for a lot of today's and upcoming problems. Yet, there are a few things that could be improved by using some paper prototyping components and their benefits. At first, paper prototypes are created to be tested by real end-users. When developers in experience prototyping try to understand existing experiences by roleplaying with team members, there might not be a single end-user involved. Although the developers collect very subjective experiences at minimum time effort, this may not be the best way to get the best understanding of the end-users' situation [17]. Hence, involving end-users more and earlier is advisable. Another critical component of paper prototyping is observing, whereas in experience prototyping, the main focus lies on "doing". A problem of experience prototyping might be that by actively experiencing, the team members may lose the overview of the whole situation and concentrate too much on subjective perception. Things that are observed by others may differ from the things someone experiences himself/herself, so additional, passive observers might have advantages.

3.2 Open Questions

How big are the benefits from using paper prototyping and experience prototyping and how big are they compared to other techniques? This question can only be answered by using several prototyping techniques for one product and comparing the results. This has been done before and it is reported in several papers, yet each product is different and results from one research may not be appropriate to another one and there will always remain a certain degree of uncertainty.

Which technique is best for which kind of interaction? This question can not be answered in detail, since each interaction is different and requires a prototyping technique that is modified accordingly for the best results. Yet, it can be answered in a more general way: paper prototyping has proven itself to be a very efficient prototyping technique for the traditional, screen-based interactions, especially in very early stages of development. Because of the limitations of the medium "paper", more hybrid interactions (like a game-controller or the "kiss communicator"), whose focus lies more on "feel" aspects of the interaction, than on the navigational part, require other techniques. Experience prototyping focuses on the users' experiences generated by products, which makes it a technique more appropriate for more hybrid interfaces. Still, this should not be seen as restriction, rather as recommendation.

4 Conclusions

In this paper, two important prototyping techniques, paper prototyping and experience prototyping, have been introduced, evaluated and compared to each other. The result was, that each technique has its place in development and that both have its benefits. Further, guerilla hci has been introduced as a method to convince those developers, who haven't been using usability engineering and prototyping in their development process, yet, to give it a try.

Lastly, I will try to give a lookout of what prototyping might be like in the future. Seeing that today's human computer interactions steadily move away from the classical, screen based interactions, paper prototyping might lose some of its importance in developing processes in the future. But the techniques that are more appropriate for the more hybrid forms of interaction still can learn a lot from it. What partially is, and will be an important fact is that using many techniques complementary increases the number of solved problems. When interactions get more hybrid, specific techniques have to be combined to a hybrid technique itself to adapt to the situation in the best possible way. The early focus on the user is already a big part in today's techniques and will also be in the future, as well as the participation of the whole development team. Creating a shared understanding of the problems is and will be crucial for solving them.

REFERENCES

[1] M. Buchenau and J. F. Suri. Experience prototyping. In *DIS '00: Proceedings of the 3rd conference on Designing interactive systems*, pages 424–433, New York, NY, USA, 2000. ACM.

[2] H. M. Grady. Web site design: a case study in usability testing using paper prototypes. In *IPCC/SIGDOC '00: Proceedings of IEEE professional communication society international professional communication conference and Proceedings of the 18th annual ACM international conference on Computer documentation*, pages 39–45, Piscataway, NJ, USA, 2000. IEEE Educational Activities Department.

[3] L. Liu and P. Khooshabeh. Paper or interactive?: a study of prototyping techniques for ubiquitous computing environments. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 1030–1031, New York, NY, USA, 2003. ACM.

[4] M. M. Mantei and T. J. Teorey. Cost/benefit analysis for incorporating human factors in the software lifecycle. *Commun. ACM*, 31(4):428–439, 1988.

[5] J. Nielsen. Paper versus computer implementations as mockup scenarios for heuristic evaluation. In *INTERACT '90: Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction*, pages 315–320, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co.

[6] J. Nielsen. Guerrilla hci: using discount usability engineering to penetrate the intimidation barrier. *Cost-justifying usability*, pages 245–272, 1994.

[7] A. Oulasvirta, E. Kurvinen, and T. Kankainen. Understanding contexts by being there: case studies in bodystorming. *Personal and Ubiquitous Computing*, 7(2):125–134, 2003.

[8] M. Rettig. Prototyping for tiny fingers. *Commun. ACM*, 37(4):21–27, 1994.

[9] J. Rudd, K. Stern, and S. Isensee. Low vs. high-fidelity prototyping debate. *interactions*, 3(1):76–85, 1996.

[10] T. Scanlon. Paper prototypes: Still our favorite. `http://www.uie.com/articles/paper_prototyping/`, 1998. visited on 10.12.2009.

[11] R. Sefelin, M. Tscheligi, and V. Giller. Paper prototyping - what is it good for?: a comparison of paper- and computer-based low-fidelity prototyping. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 778–779, New York, NY, USA, 2003. ACM.

[12] K. T. Simsarian. Take it to the next stage: the roles of role playing in the design process. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 1012–1013, New York, NY, USA, 2003. ACM.

[13] C. Snyder. Using paper prototypes to manage risk. `http://www.uie.com/articles/prototyping_risk/`, 1996. visited on 10.12.2009.

[14] C. Snyder. Paper prototyping. `http://www.cim.mcgill.ca/~jer/courses/hci/ref/snyder.pdf`, 2001. visited on 10.12.2009.

[15] C. Snyder. *Paper prototyping: the fast and easy way to design and refine user interfaces*. Morgan Kaufmann, 2003.

[16] J. M. Spool. Looking back on 16 years of paper prototyping. `http://www.uie.com/articles/looking_back_on_paper_prototyping/`, 2005. visited on 10.12.2009.

[17] D. Svanaes and G. Seland. Putting the users center stage: role playing and low-fi prototyping enable end users to design mobile systems. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 479–486, New York, NY, USA, 2004. ACM.

[18] R. A. Virzi, J. L. Sokolov, and D. Karis. Usability problem identification using both low- and high-fidelity prototypes. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 236–243, New York, NY, USA, 1996. ACM.

[19] T. Z. Warfel. First prototyping survey results. `http://www.rosenfeldmedia.com/books/prototyping/blog/first_prototyping_survey_resul/`, 2008.

[20] wikipedia.com. Paper prototyping. `http://en.wikipedia.org/wiki/Paper_prototyping`, 2002. visited on 10.12.2009.

From paper prototyping to sketching with hardware

Melanie Kunz

Abstract— In this paper, I describe the common paper prototyping with their pros and cons and compare them with future prototyping techniques. To make a work process more efficient and enjoyable, the job of experience prototyping is to find user needs, given from experience and help to generate new ideas. It is designed to understand existing user experience, explore and evaluate design ideas or communicate what be like to engage with. With paper prototyping you can visualize the interface, which is manipulated by person, very fast.

Index Terms—paper prototyping, experience prototyping, prototyping, design, methods

1 Introduction

A bordered representation of a design where users can interact is a prototyp. So prototyping is a method of software engineering, which show you first results and early feedback. To get this, you design a view executable paradigm of the final product with growing functions and present it the client. A prototype is also the first full-size model to be manufactured and has the essential features.
But why do we need prototypes? You can check the usability and the producibility very early in the process. Also you can validate the customer requirement with a prototype.
You can split prototyping in Low- and High-Fidelity Prototyping. Low-Fidelity prototyping is the visualization of design ideas at very early stages of the design process. The development is very simple and does not need very much time [7]. In contrast, high-fidelity prototypes are fully interactive. Users can enter data in entry fields, respond to messages, select icons to open windows and interact with the user interface as though it were a real product [6]. To certify the idea and the usability of the product it is better to use the Low-Fidelity prototype because the short time to make it. To build and change a High-Fidelity Prototype takes a long time *(see figure 1)* and is more expensive. This is more to check details like visualization and functions.

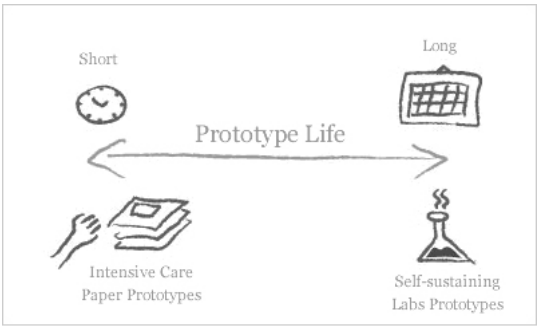


Fig. 1. The life of a Prototype: Low- and High-Fidelity Prototyping [1]

To cut down on the complexity of implementation by eliminating parts of the full system you can split prototyping vertical and horizontal *(see figure 2)*. Horizontal prototypes reduce the level of functionality and result in a user interface surface layer[4]. This means, that the prototype has all features, but not with the whole functionality. In contrast vertical prototypes reduce the number of features and implement the full functionality of those chosen [4].
You can see the differences on an example. Think about a simple mobil phone with the three features: calling, messaging, photographing. With a horizontal prototype we could call, send messages and photograph. But we cannot use all functions. Maybe we can only call one person or we can only make colored pictures. The function to make black-and-white pictures isn't implemented. A vertical prototype could have only the feature of messaging. But now we can use all functions of messaging and send everybody a message. It can be a person of the contact list or only a phone number. We can send sms and mms.

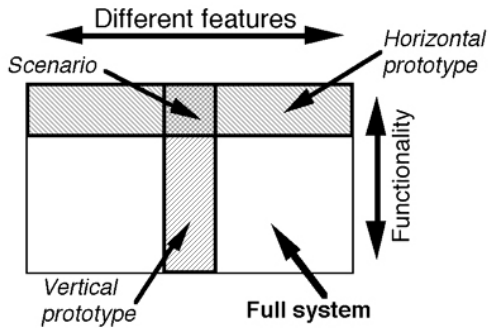


Fig. 2. The concept of a scenario compared to vertical and horizontal prototypes as ways to make rapid prototyping simpler [4]

2 From paper prototyping to sketching with hardware

This section will describe paper prototyping and experience prototyping, one part of sketching with hardware. At the end of this part I will compare this methods.

2.1 Paper prototyping

In this capture I will show you the history and definition of paper prototyping. After this introduction you can see how it works and what pros and cons we can find.

2.1.1 History

Paper prototyping started in the mid 1980s. In the early 1990s it was a fringe technique, used by a few pockets of usability pioneers but unknown to the fast majority of product development teams. But by the mid-1990s, paper prototyping was catching on and became popular when companies such as IBM, Honeywell, Microsoft, and others started using the technique in developing their products. These companies experimented with the technique and found it useful and started using it as an integral part of their development process. The

- *Melanie Kunz is studying Media Informatics at the University of Munich, Germany, E-mail: Melanie.Kunz@campus.lmu.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009*

concept of ”low-fidelity” prototyping popping up about 1990 from authors like Jakob Nielsen, Bob Virzi and Tom Tillis. Today paper prototyping is at many companies a mainstream practice, but there are although many people who’ve never heard about it. [10].

2.1.2 Definition

- *What is paper prototyping*

  Paper Prototyping is a widely used method for designing, testing and refining user interfaces. It is a variation of usability testing where representative users perform realistic tasks by interacting with a paper version *(see figure 3)* of the interface that is manipulated by a person ”playing computer”, who doesn’t explain how the interface is intended to work [10]. It is useful for Web sites, Web applications and conventional software. Paper prototyping is low-tech, so you can get maximum feedback for minimum effort.

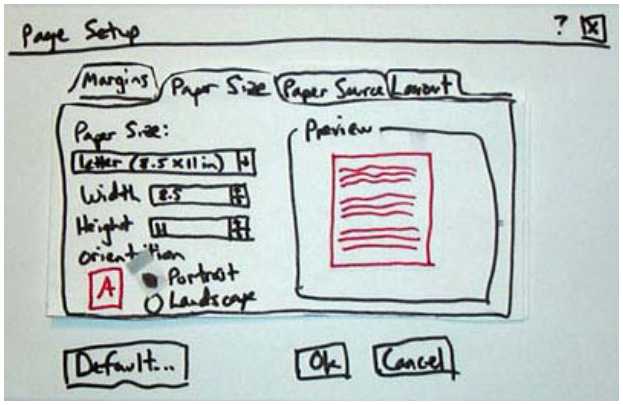


Fig. 3. A paper prototype of a File Setup dialog from Microsoft Word [9]

- *How it works*

  First you have to decide with other members of your product team the type of the representative user and on the tasks that you’d like this user to accomplish. If you know your target audience and the tasks, you make screen shots and/or hand-sketched drafts of all the windows, menus, dialog boxes, pages, popup messages and so on that are needed to perform those tasks. *Figure 3* shows an example of a paper prototype of a file dialog from Microsoft Word.

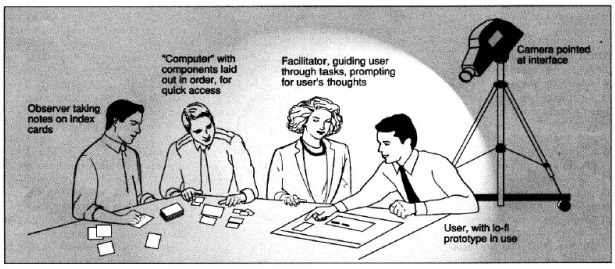


Fig. 4. A low fidelity testing session [5]

  To create the prototype, it isn’t necessary to have a working version of the interface. Your product team has to know which features wish to be checked. You only need to create all features you need for this task. After creating the prototype, you have to conduct a usability test with a person who is representative of the audience. The user has to interact with the prototype directly and he has to think aloud. So you can see where missunderstandings are and which parts of the interface are selfexplanary or confusing. One team member has to play the role of ’computer’ while the user ’click’ by touching the prototype buttons or links and ’type’ by writing their data in the prototype’s edit fields. This ’computer’ has to simulate how the interface behaves and doesn’t explain how it is supported to work. A faciliator conducts the session while other members of the development team observe and take notes [9] *(see figure 4)*.

- *How good should the prototype be*

  For your prototype you can chose screen shots and/or hand-sketched drafts. Because the prototype is all on paper, you can easily change some parts of the interface during the usability test. So it is not necessary to have a very nice prototype. The prototype only needs to be good enough for you to get answers to the questions you’re most concerned about. You don’t need straight lines or typed text. You want to find out whether your prototype is selfexplanary and not if it’s nice to look. You are allowed to tell the user what a word is, if he can’t read it. But it isn’t allowed to explain him a word or sentence if he can not understand it. The look of the paper prototype has not to be lovely, so it is enough to use words instead of images or icons and don’t use colors. The goal of low-fidelity prototyping is to get a prototype very fast. Very often, the first usability test will show you problems you’d never anticipated and then you’ll want wo make changes. So don’t spend to much time making the prototype look neat. If it’s legible, it’s good enough. [9].

2.1.3 Pros and cons

Some development teams use paper prototypes in the early stages of designing. It is on the one hand especially useful for gathering data about some problems, but on the other hand it isn’t ideal for bigger questions. Carolyn Snyder conducted in July 2002 an online survey of some usability professionals about their use of paper prototyping and their attitudes toward it. 172 people - most of them usability specialists - attended to this survey. To the question ’What is the importance of paper prototyping to your work’ 86% answered either ’essential’ or ’useful’ *(see figure 5)* [10].

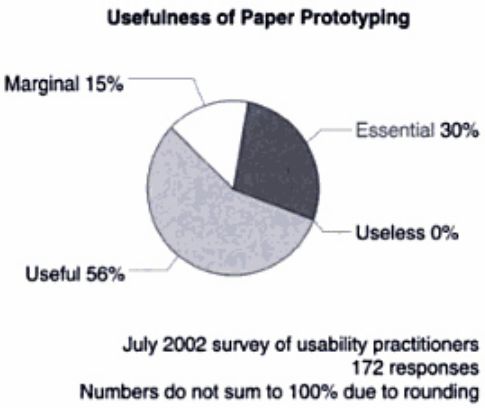


Fig. 5. Answers the quesion, ’What is the importance of paper prototyping to your work?’ from a July 2002 survey of usability professionals [10]

Their are a lot of pros of paper prototyping, but also there are some combatants. In this section I will show you the assets and drawbacks.

- *Validity - Does paper prototyping find real problems? Does it find the same problems as testing the real interface?*

  Validity is probably the biggest concern people have - they're afraid of that because a paper prototype doesn't look realistic, the results from usability testing aren't realistic either.
  In a study of Reinhard Sefelin, Manfred Tscheligi and Verena Giller, four prototypes of two systems were developed [7]. One system was a calendar system, which enabled users to enter meetings, classes, birthdays and anniversaries. Its user could request his entries in a daily and in a weekly overview. The second system was a touch screen ticket machine, which enabled users to buy tickets and to request information concerning their journey and concerning discount packages. For both systems a computer-based and a paper prototype were built.

	Critiques			Suggestions		
	Mean	Std. Dev.	t-test	Mean	Std. Dev.	t-test
Ticket machine	8.96	5.49		6.96	4.33	
Computer prototype	9.25	6.96	t= -.25 p= .801	7.08	5.43	t= -.14 p= .891
Paper prototype	8.67	3.80		6.83	3.10	
Calendar system	10.17	5.40		7.83	5.48	
Computer prototype	9.25	4.14	t= -.82 p= .418	6.42	4.50	t= -1.28 p= .213
Paper prototype	11.08	6.49		9.25	6.18	

Fig. 6. Total number of critiques and suggesions [7]

Figure 6 shows that the number of critiques and suggestions is not affected by the kind of prototype. The table shows also the results of t-tests for two independent samples, which also did not show significant differences. Paper- and computer-based low-fidelity prototypes lead to almost the same quantity and quality of critical user statements.

In a study from Virzi, Sokolov and Karis (1996) two experiments are made. In the first experiment the probant used a CD-ROM based electronic book and in the second he used an interactive voice response system. One group performed the tasks with a paper-based low-fidelity prototype, while another did the same with a high-fidelity prototype or the actual product. Both experiments showed that users were able to do the same tasks, in about the same number of steps, regardless of the kind of prototype. These results imply that if something is hard to do with the paper prototype, it's probably hard to do with the real interface as well.
The study from Novick (2000) don't prove that a paper prototype will find the same set of problems as a more realistic one. Rather, they indicate that the sets of problems overlap to a considerable degree and there don't appear to be important differences in the problems that a found with one method versus another [10].

- *Bias - Does paper prototyping introduce false problems? Does it change users' behavior or feedback in such a way that we can't trust the results?*

  The study with the calender system and the ticket machine of Reinhard Sefelin, Manfred Tscheligi and Verena Giller showed also that subjects confronted with paper prototypes show a greater willingness to draw their suggestions [7].
  Sometimes a seemingly free-floating anxiety about the results from paper prototype test is valid but caused by something other than the prototype. It is not cogitable that the user find different problems [10].

- *Professionalism - What will other think of this technique (and us for using it)? Will the prototype be perceived as sloppy or amateurish?*

  The study of Kitchen-Net supports the task of working in an industrial kitchen by responding to spoken queries for items. Kitchen-Net uses a set of screens placed around the kitchen. The screens nearest the user shows directions to his or her requested items. For the paper version they used paper sketches and post-its, for the interactive prototype they replaced the paper screens with Vadem Clio Handheld PCs, so the prototype automatically responded the events. The results of this study is, that the designers give more flexibility in the early phases, but paper prototypes are insufficient for formal user studies because of their validity issues and need for more staff to support interaction over long periods of time and large locations [3].
  The study of Reinhard Sefelin, Manfred Tscheligi and Verena Giller showed that subjects prefer computer prototypes. Since the comfort of subjects is one of the major factors of a successful usability test [7].
  It is natural for developers to be concerned that others will perceive their work as unprofessional if all they see is a sloppy and incomplete paper prototype. Eyeryone has his own opinion.

- *Resources - Do we have time for this? Is there a payoff here, or is this just extra work? Why not just wait until the real thing is ready?*

  For this question you have to decide by yourself. A paper prototype is very easy and fast to build. You need about one day to create a paper prototype, in contrast you need some weeks to implement it. If most of your questions can be answered by a paper prototype, it isn't extra work. You can see what the problems and missundertandings are.

2.2 Experience prototyping

In this section I will describe experience prototyping. It will start with the history and the definition. After this I will show you experience prototyping in practice and pros and cons.

2.2.1 History

The earliest recorded use of role playing and low-fidelity prototyping in the design of computer systems dates back to the UTOPIA project in the 1980s [11].The term interaction design was first proposed by Bill Moggridge and Bill Verplank in the late 1980s. To Verplank, it was an adaptation of the computer science term user interface design to the industrial design profession. To Moggridge, it was an improvement over soft-face, which he had coined in 1984 to refer to the application of industrial design to products containing software. In 1989, Gillian Crampton-Smith established an interaction design MA at the Royal College of Art in London (originally entitled "computer-related design" and now known as "design interactions") [12]. Experience Prototyping is very new. It comes more and more famous.

2.2.2 Definition

- *What experience prototyping is*

  'Prototypes' are representations of a design made before final artifacts exist (see Introduction). Experience depends upon the perception of multiple sensory qualities of a design, interpreted through filters relating to contextual factors. An Experience Prototype is any kind of representation, in any medium, that is designed to understand, explore or communicate what it might be like to engage with the product, space or

system we are designing. You can find user needs, given from experience and help to generate new ideas [2].In addition Brandt and Grunnet describe the use of role playing as a way for designer and users to have a dialogue about design ideas [11]. With experience prototyping you can get an idea from first hand with an active contact to the prototype.

- *Why experience prototyping is important*

  Experience prototyping is a good way to explore, communicate and interact with the designers we develop like experiencing cycling on the ice, although the mood, snow conditions, bicycles type and many other factors really matter and tend to change time [13].
  It becomes more and more important to design complex and dynamic interactions with converging hardware and software, spaces and services - products like a digital camera *(see figure 7)* [2].

Fig. 7. Digital camera interaction architecture prototype [2]

For this the designer needs to focus an 'exploring by doing' and actively experiencing the sometimes subtle differences between design solutions. Therefore, it is a powerful asset to have tools and techniques which create a shared experience, providing a founding for a common point of view. Experience prototyping allows us to angage with new problems in new ways.

2.2.3 Experience prototyping in practice

We have identified three different kinds of activities within the design and development process where Experience Prototyping is valuable:

- *Unterstanding existing user experiences and context*

  The first step is to understand user experiences and context. You have to share understandings from the field, recreate observed situations or create extrapolations based on an understanding of the observations [8]. Experience Prototyping is applied to demonstrate context and identify issues and design opportunities.

  One opportunity is the patient Experience. You can create essential experience elements with the imaginations of the people. The participants translated their own experiences into patients' needs. For example they appreciated the importance of warning information to help patients anticipate and prepare for a shock. They also saw the need to provide information to indicate the patient's condition to bystanders, and a broader base of remote support for this next generation of products and services. This is necessary that you can simulate important aspects of the real user experience [2].
  Another example is the ROV Pilot Experience. It used a proxy device to provide the team with specific insight into an experience that was not readily available to them [2].

  Another opportunity to understand existing experiences you can see by workshop of Dag Svanaes and Gry Seland. It was to explore potentials for new mobile devices for teenagers, looking 5-10 years ahead. The first session of this workhop was the scenario development through a drama. The teenager all were able to use drama as a technique to stage scenarios from their own lives [11].

- *Exploring and evaluating design ideas*

  In this step the development team has to find a solution how to create an experience prototype. Normally you build your prototype with easy things you can find and convert them.

Fig. 8. Control in an immersive video environment [2]

An example for Controller for an immersive environment are a tactile immersive experience represented by a palm-sized pebble. If you want to controll with both handy and split the control functions, you can take two different-sized joysticks mounted on suction pads. To move the controller with your full-body, you can let your probant sit down on a skateboard [2]. Using everyday items makes it very easy to test few opportunities for a scenario and the subject can valuate *(see figure 8)*.

Another example is Experiencing an Airplain Interior. This is completly different to design becuase this user experience was designed in public environment.

Fig. 9. Bodystorming layouts for an airplane interior [2]

Figure 9 shows a variety of bodystorming explorations within an environment simulating inside of an airplane. The team enacted for example with chairs various social situations. Many ideas of physical configuration could be tested in a time and money efficient manner. Ideas were generated and evaluated rapidly the team as they directly experienced physical and social issues in this full-scale environment [2].

- *Communicating ideas to an audience*

  The practice of creating physical performances to communicate developed ideas, issues and scenarios to an audience, is the last step. Informances might also be used in any design phase to convey current ideas and issues in a rich way [8]. Here you can see wheter your idea is intuitionally. The last step is very important to get more arguments.

2.2.4 Pros and cons

It is known that people rarely use the recommended usability engineering methods. You can find three basic techniques. These are an early focus on the user, an empirical measurement and an interactive design.

What should we do when developing and evaluating a new computer system for end users? It is very sad that only 6% of the people used all principles - as a study of Gould and Lewis proved 1985. But why is it so little-known? One possible reason are the costs of using the techniques. The most people think it's not lucrative to spend money in it. Another reason is, Human-computer interaction (HCI) methods are seen as too time consuming and because the techniques are often intimidating in their complexity [4].

But their are still many benefits. One great advantage of Experience Prototyping is that it requires hybrid and overlapping skill-sets such that it is not exclusive to any single design discipline. As such, it offers an opportunity for all types of designers to supplement their traditional discipline skills in an effective and broadening way [2]. Experience prototyping also simulates important aspects of the whole or parts of the relationships between people, places and objects unfold over time.
You can provide inspiration, conformation or rejection of ideas based upon the quality of experience they engender. By enabling others to engage directly in a proposed new experience it provides common ground for establishing a shared point of view [2].

Experience prototyping is not about the creation of a formalized toolkit or set of techniques, but is about developing an attitude and language to solve design problems.

2.3 Comparison of paper and experience prototyping

In this section I will compare paper and experience prototyping and skow how to improve prototyping.

- *Compare the paper prototyping with the new method experience prototyping*

  Prototyping provides always user feedback early in the process. This has the benefit, that you can change your modell before you have invested effort in implementation. Also you can experiment with many ideas wather than betting the farm on just one. Between the development team and the customer you can communicate and attach their ideas contemporary. Paper and experience prototyping share a lot of benefits [10].

  Whether paper prototyping is good for questions about concepts, terminology, navigation, content and functionality, it isn't not ideal for technical feasibility. Experience Prototyping is a newer technique and comprise more complex software or hardware.
  Paper prototyping comes to the fore the functionality, on the other hand for experience prototyping the design is interesting. But both techniques want to be self-explanatory.

- *Show, how to improve prototyping*

  The disadvantages about prototyping are basically that you cannot find all and mainly the right problems. These problems have to be reduce more and more, to get a fast designed prototype with same findings like the real world. To improve prototyping and enlarge the options, newer techniques like experience prototyping are a good way. They allow more options in experience. What the future will bring we don't know, but the main goal of prototyping will still be maximum feedback for minimum effort.

3 Discussion

In my opinion prototyping is a very good idea. It makes it possible for less technic dedicated persons to help by the development. They can be creative with new ideas without the know-how about programming. Also you can design very fast a prototype. So if you find out, that your idea is maybe no user need, you didn't lose too much time. Anyway I think, that it isn't indispensable to make a usability test with your implemented version. You cannot find always all problems with a prototype, so you can only be on the sure side, if you integrate the customer often in the development process.

The main pro of prototyping is the big flexibility. The programmer has not to make a static structur. So you can change parts of your prototype during the usability testing. This makes prototyping very interesting. But there are a lot of prototype iterations needed and this makes the control more different.

Furthermore it is a bad idea to make always a prototype. Sometimes it is very easy to implement an easy version of the scenario. Then you spent time in a prototype for nothing. Therefore the main thing is, to prove first. You have to know what you want and how to do it.

4 Conclusion and future work

Prototyping, especially the new techniques like Experience prototyping are at the moment almost unexplored and contentious. There are some disputants, but also more and more follower. Disputants think that a prototype is only a waste of time and the results aren't the same as the results in the real world. For follower it is a very easy and cheep opportunity to find real user needs before the implementation.

In future the technique will grow up more and more. I think paper prototyping will stay a good opportunity to get maximum feedback for minimum effort. But for more complexer hardware or software other techniques will be needed. Maybe experience prototyping will become more famous when popular concerns show how it works. But therefore more studies will be needed.

References

[1] Sustainable ui prototyping. http://linowski.ca/thoughts/category/sketches/, 2009. accessed 04-December-2009.

[2] M. Buchenau and J. Suri. Experience prototyping. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 424–433. ACM, 2000.

[3] L. Liu and P. Khooshabeh. Paper or interactive?: A study of prototyping techniques for ubiquitous computing environments. In *CHI'03 extended abstracts on Human factors in computing systems*, page 1031. ACM, 2003.

[4] J. Nielsen. Guerrilla HCI: Using discount usability engineering to penetrate the intimidation barrier. *Cost-justifying usability*, pages 245–272, 1994.

[5] M. Rettig. Prototyping for tiny fingers. *Communications of the ACM*, 37(4):21–27, 1994.

[6] J. Rudd, K. Stern, and S. Isensee. Low vs. high-fidelity prototyping debate. *interactions*, 3(1):85, 1996.

[7] R. Sefelin, M. Tscheligi, and V. Giller. Paper prototyping-what is it good for?: a comparison of paper-and computer-based low-fidelity prototyping. In *Conference on Human Factors in Computing Systems*, pages 778–779. ACM New York, NY, USA, 2003.

[8] K. Simsarian. Take it to the next stage: the roles of role playing in the design process. In *Conference on Human Factors in Computing Systems*, pages 1012–1013. ACM New York, NY, USA, 2003.

[9] C. Snyder. *Paper prototyping*. Morgan Kaufmann, 2003.

[10] C. Snyder. *Paper prototyping: The fast and easy way to design and refine user interfaces*. Morgan Kaufmann Pub, 2003.

[11] D. Svanaes and G. Seland. Putting the users center stage: role playing and low-fi prototyping enable end users to design mobile systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 479–486. ACM New York, NY, USA, 2004.

[12] Wikipedia. Interaction design. http://en.wikipedia.org/wiki/Interaction accessed 04-December-2009.

[13] A. Yasar. Enhancing experience prototyping by the help of mixed-fidelity prototypes. In *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, pages 468–473. ACM, 2007.

Prototyping for Web Interfaces

Gerald Beck

Abstract— Prototyping for web interfaces is a task that brings together various actors. There is a wide range of tools for prototyping but not all of them meet the needs of the involved actors. This paper presents four tools for prototyping: paper prototyping, PowerPoint, DENIM and WARP. It shows the advantages and limits of the tools and of prototyping for web interfaces. To support the use of prototyping, the paper suggests that with the help of classifications and catalogues of prototyping tools, prototyping might become more relevant for web interface design practice because designers will have it easier to find tools that fit their needs.

Index Terms—Prototyping, Web Interface, Design Process, Graphic Design, Classification

1 Introduction

Software projects for the WWW involve a number of actors with heterogenous skills, roles, backgrounds and expectations. Depending on the complexity of the project, software developers, database experts, graphic designers, content providers, clients and potential users contribute to the development of well designed web interfaces. These actors need to be involved in different stages of the design process. The visual designers for example have to be involved at an early stage to contribute their idea of the interface [2]. As graphic designers are not necessarily trained as computer scientists, they need tools for prototyping that meet their way of working and need less programming skills. This paper focusses on the role of graphic designers and analyses tools for prototyping from their perspective. At the same time it is clear that the other actors named above have specialized requirements to prototyping tools as well. They are important as well but cannot be covered in this paper as prominent as graphic designers.

Tools for prototyping are usually classified due to the fidelity of the prototypes they produce. Taking the example of the special needs for graphic designers as an point of departure, alternative ways of classifications might be more useful. This becomes obvious when we ask for the practical use of classifications. When designers choose from the vast amount of tools available, a practical classification can help to narrow the choice and save evaluation resources.

Against this background, this paper will ask about the role of prototyping tools in the design process of web interfaces and the use of classification schemes. The second section will locate prototyping in the design process of web interfaces and will have a look at the benefits of prototyping for web interfaces. The third section will describe four different tools for prototyping of web interfaces. In the fourth section we will discuss alternative ways to categorize prototyping tools.

2 Prototyping for Web Interfaces - Why and When?

This section will explain why prototyping for web interfaces is important, why special tools are required and who will use the tools in which phase of the design process. To define what is meant by prototypes we will follow the practical definition of Bochicchio et al. who see prototyping as "fast, cheap and reliable development of a mockup of the final application" [2].

2.1 Prototyping Will Improve The Design Process

Why should designers of web interfaces use prototyping in their work? Prototyping means to invest valuable resources in something that will not be the final result of a project. This can only be justified if projects that invest in prototyping are likely to produce better (and faster) results than projects that have a straight forward strategy and invest only in the final system from the very first day. Empirical proofs are hard to find here but a lot of scholars advocate for prototyping with good reasons.

As introduced above, prototyping can be valuable in the design of web interfaces because of the possibility to get feedback from the project participants and users in a very early stage of the design process. The prototypes for this purpose do not have to be high fidelity. Low fidelity prototypes proved to deliver as good results as Virzi et. al. showed for the identification of usability problems [13].

Usability testing with prototypes can make web development a lot cheaper than testing with final versions of web interfaces. Because these tests will always show difficulties that have to be corrected. If a test shows these problems before the actual development of a system, the search space for solutions will be a lot larger, because the design is not already locked in a certain way of development. Thus less work will be lost during the project and the solutions can be more innovative.

Lim et al. [4] mention an other strength of prototyping. Not only is prototyping a good way to identify usability problems of web interfaces, it is a "means by which designers organically and evolutionarily learn, discover, generate, and refine designs". This means, they do not only help to avoid mistakes and wrong design decisions but they enable designers to generate better ideas and to find alternatives to given paths of development. Thus the strength of prototypes goes beyond the task of securing requirements of web interfaces. They can become a medium of communication for discussing various design solutions without really developing the discussed versions [2].

Prototypes for web interfaces can work as a translation medium in heterogeneous projects. Clients will read their text based requirements for the system to be developed in an other way than software engineers or visual designers will. Prototypes visualize their ideas and are a chance for early interventions from all group members.

Prototyping should support the early stage of the design process where design ideas have to be evaluated, tested and discussed in a reflexive and iterative way. Another striking reason is the variety of content options that cannot be implemented by automatic tools. Thus automatic implementation is in danger of favoring some forms of content over others just for technical reasons that are not based on design decisions [2].

Some prototyping tools offer export functions that make them useful for supporting the implementation of web interfaces rather than for the mere design process. The question is, whether such tools are still flexible enough to support the discussions on design that is the central task of prototyping tools. Bochicchio et. al. are critical about this question and name several reasons why tools should focus on supporting design and prototyping and not on implementation.

2.2 Who will be involved in prototyping?

During the design process of a web interface two groups of actors are most likely to actually do the prototyping and thus use tools for prototyping: the graphic designers and the software developers. As web interfaces become more and more visual, the graphic designers might have a special need for prototyping. Their ideas are the basis

- *Gerald Beck is studying Media Informatics at the University of Munich, Germany, E-mail: @campus.lmu.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009*

for further development by the other actors. Graphic designers are not necessarily trained in programming and will have problems with tools for prototyping that require a lot of formal scripting. They would rather use tools that fit to their intuitive and iterative way of designing for web interfaces.

Prototypes can be a bridge between graphic designers and software developers. The process of prototyping makes them see the project in a common way and will render visible possible problems concerning the usability of the designed interface. A refined prototype on the other hand can be a very valuable model on which software designers can base their programming and graphic designers can base their visual work.

There can be more groups involved in the design process of web interfaces. Often enough users are a forgotten group as it is not easy to define target groups for web projects upfront. Nevertheless users are an important group and should be integrated in the design process. User integration can be more than usability testing. Studies show that discussions with users do not only show usability problems but can generate new and creative solutions [12].

3 Tools for Prototyping

Prototyping for web interfaces can improve the design process as we have seen in section two. There is a wide range of tools to support prototyping of web interfaces and it is not always easy to find and choose the right tool for the right task. To give examples of tools and methods that support the prototyping process of web interfaces, four different approaches will be presented:

- Paper prototyping (section 3.1) as a classic and easy to apply method that can produce rapid prototypes that are suitable in the early stage of a project when it is not necessary to have fine graphics on screens [10].
- A ”mid fideltiy” prototyping framework that relies on Powerpoint as a tool that can be easily used without programming skills [3] (section 3.2).
- The DENIM system as an example of a sketch based visual language that can be used in the early stages of design and supports HTML export of generated web interface prototypes [6] (section 3.3).
- As a fourth example the WARP environment (Web Applications Rapid Prototyping) will be presented in section 3.4. It has been chosen as an example for online software tools for prototyping web interfaces [1].

The tools are chosen in order to cover the range of available tools in respect to the effort to concentrate on tools that focus on design prototyping rather than implementation. Nevertheless a lot of more or less sophisticated approaches are available that cannot be presented in this short paper. For a overview of alternative tools see for example the work of Bochiocchio and Fiore 2004 [1]. There are also tools that cover the entire design process from requirement definition to the coding of html pages. The environment for rapid prototyping of web applications of HyperDe, introduced by Nunes and Schwabe is one example [**?**].

3.1 Paper Prototyping

Paper prototyping is a very rapid way to create prototypes for web interfaces. ”Paper prototyping is a variation of usability testing where representative users perform realistic tasks by interacting with a paper version of the interface that is manipulated by a person playing computer, who doesnt explain how the interface is intended to work.” [11]. This definition of paper prototyping focuses on its potential in user interface testing. Studies showed, that paper prototypes are as good for testing user interfaces as computer based prototypes [10].

Their study also showed, that subjects give the same quality and quantity of suggestions, no matter if the prototype is paper based or computer based. But subjects prefer computer based prototypes. Nevertheless they find good reasons for paper prototyping. First, paper prototypes are more flexible and will support any idea of a user interface whereas computer based tools might limit the creative space of designers.

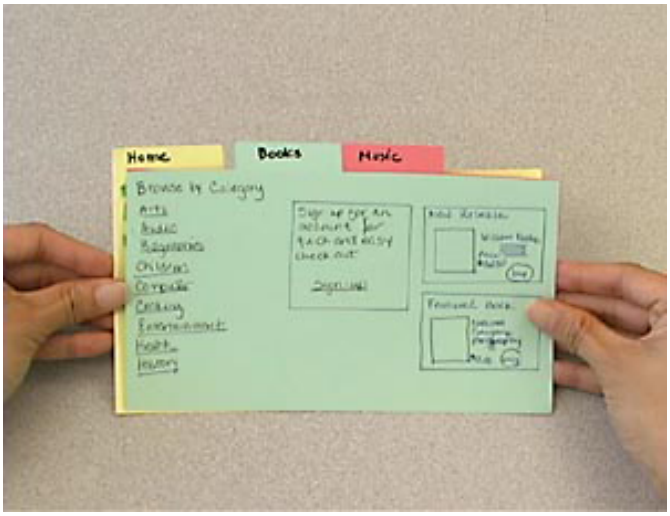


Fig. 1. Paper Prototyping [9].

Second, paper prototypes do not ”exclude members of the design team without sufficient software skills” [10]. These actors might be for example the graphic designers as mentioned above. This brings us to the third reason mentioned in the study. Paper prototypes have a strength if the user interfaces are dominated by visual elements because they can be drawn and redrawn very quickly during discussions of the design team.

3.1.1 Usability Testing with Paper Prototyping

How will a usability test supported by paper prototyping work? Snyder [11] describes the praxis of paper prototyping as a workshop setting with multiple people. The design team prepares screenshots or drawings of all relevant parts of the interface, including pop-up windows, mouseover events, etc. This method is very cheap, because nothing else than paper, pens and scissors are needed *(see figure 1)*.

In the workshop setting described by Snyder, she identifies four groups of participants: user, faciliator, computer and observer. Usually the members of the team take the role of the computer or the observer. The computer has to act according to the actions of the user without explaining why it performs like it does. The faciliator can be a user interface expert. The users are asked to interact with the prototype. Any difficulties that appear during the test will be documented by the observers and have to be subject of discussion about improvements of the prototype in the next phase of the development.

3.1.2 Why Paper Prototyping?

We have seen that paper prototyping can be very useful in the early stages of the design of web interfaces. It can be used for usability tests and for improving the (visual) design process. Paper prototyping has the advantage that no programming skills are necessary for designing the prototypes. The design can be very flexible and is not limited to the capability of a software tool. Thus paper prototyping is a very cheap, flexible and fast way to design, improve and test user interfaces for web applications.

There are certainly limits of paper prototyping. Changes in the interface design will produce a lot of effort in paper prototyping. More than in most computer based tools that give the opportunity to change, for example the navigation structure of a web interface for the hole system in one design step. A second limit is the poor visual fidelity of paper prototyping. The result is far away from what users will see as an end product. This requires a high grade of imagination which

cannot always be assumed, especially in commercial projects with demanding customers.

3.2 PowerPoint as a Tool for Prototyping Web Interfaces

Before we will introduce the DENIM framework, that has some similarities with paper prototyping, we will have a look at a software that has been designed for presentation but can be very useful for prototyping web interfaces. Web sites are generally made from little boxes. These boxes can contain text, graphics, hyperlinks, froms, etc. [7]. For rapid prototyping after the very early steps of the design process, Engelberg and Seffah propose a "mid fidelity" framework that is based on MS-PowerPoint [3]. The methods proposed by the authors for MS-PowerPoint can be performed with any other presentation software like Apple Keynote or OpenOffice as well.

The intention of this rapid prototyping approach is to "prototype the interactive and navigational aspects of user interfaces as quickly as possible, with a minimum amount of investment of learning time and with no programming skills" [3].

The authors are critic towards the simple distinction between low and high fidelity tools. Their critique is based on the fact that the definitions of the two extremes are weak and that there can be very different outputs from two tools that are both classified as "low fidelity". This critique on the classification scheme will be elaborated in the fourth section. Now it is only mentioned because it lead the authors to call their approach "mid fidelity". The characteristics of mid fidelity tools can be summarized as follows [3]:

- used after early design
- used for detailed design and usability evaluation
- detailed information about navigation, functionality, content and layout in approximate form

The basis of the prototyping process according to Engelberg and Seffah is a textual outline of the content in tree format and the definition of the visual pattern for the overall design. In their method, they distinguish six steps in the process of mid fidelity prototyping.

3.2.1 Define Screen Capabilities and Font Size

One of the first decisions in the design process of web interfaces is the definition of the screen size. Usually a 800x600 pixels or a 1024x768 pixels resolution is chosen. With growing sizes of monitors, this resolution might shift to larger sizes in the future. The font size depends on a) the amount of textual content that should be shown, b) the length of the items in the menues and c) the estimated user group.

PowerPoint will not natively support this step because the full-screen modus is still optimized for 800x600 pixel. So the font size has to be altered. For 800x600 pixel the authors suggest 12-14 pt and for 1020x768 they suggest 10-12pt fonts [3].

3.2.2 Define Main Areas and Navigation

The main areas and the navigation of a web interface will appear on all pages and sub pages. In this step, the have to be defined from the overall design. It is suggested to save these elements to the slide master of PowerPoint. The slide master will make the main areas and the navigation appear on every new page. If the design team decides to change for example the navigation structure of the interface or the position of a main area, the changes can be easily made in the slide master and will at once apply to all pages of the prototype.

The master can only be used for elements that do not change if users go from one page to the next. It can be used for headers and background graphics as well as for navigation. But it cannot be used for dynamic expanding navigation.

3.2.3 Homepage-Design

The design of the homepage is very relevant for the design process as it will be the basis of the look and feel of all other pages. Even in prototypes it is worth investing in this step as it helps users and clients to imagine how the rest of the page might look like.

PowerPoint supports integration of graphics but is not that valuable in creating refined graphics. In this step the graphic designers are required to provide versions of possible homepage designs.

3.2.4 Insert Pages and Links

Now the pages can be inserted. The slide master will help to avoid repeating work. Thus only new items for sub pages have to be inserted. The slideshow mode of PowerPoint can help to simulate links and navigation.

It is possible to insert links that link to other slides within the presentation by using anchors. The easiest way is to use the page title as an anchor as it is not necessary to define anything on the target page. Engelberg and Seffah suggest to copy and paste repeating links like the navigation [3].

3.2.5 Testing

The use of links avoids that users are locked into the hierarchical structure of the slideshow. They can click on unexpected links, move backward and forward and thus make usability problems visible.

3.2.6 Refine Page Contents and Links

After the raw setup of the prototype is made it can be refined and detailed. PowerPoint supports a lot of features but the larger the project gets, the more complicated it will be to alter navigation structures and links as there is no feature to manage overall links in powerpoint.

Still Engelberg and Seffah provide a method that uses an easy to learn tool that most subjects that are involved in the design process of web interfaces are familiar with. No programming skills are needed and changes in the prototype can be made right in the tests or design sessions.

3.3 DENIM: Computer-Based Sketching

The DENIM framework is a visual language for sketching large and complex interactive designs [6] [5]. It is a tool that combines some elements of the two approaches mentioned above. Like paper prototyping, DENIM relies on the designers ability and wish to sketch during the design process. The framework supports pen based computer interfaces and sketching throughout the prototyping process. Like PowerPoint, DENIM is computer based and allows users to navigate the interface on a computer screen.

The developers of DENIM rely on two ethnographical findings [8] about the work of web designers. First, web designers sketch on paper during the early design process. So it has been of central interest that DENIM supports sketching. Second, it can be distinguished between three levels of sketching: site maps, storyboards and individual pages. DENIM gives the opportunity to bring these three levels together by zooming.

3.3.1 How to use DENIM?

The main part of the DENIM user interface is an infinite canvas on which the designer can sketch the content of pages. The links between pages are drawn as arrows *(see figure 2)*. The zoom interface provides five levels of zooming: overview, site map, storyboard, page and detail. On the bottom of the screen, a toolbox that contains the tools for drawing, panning and erasing is positioned [5]. The designer can draw pages on the canvas and connect them with arrows. There are two kinds of arrows: organizational and navigational. Organizational arrows are created to show general connections between pages or supposed paths that users might take interacting with the interface.

Navigational arrows specify "a transition from one page to another" [5]. They represent hyperlinks on web pages. Designers can draw the arrows in DENIM. Orgnizational arrows go from one page to an other. Navigational arrows lead from a specific object on one page to the next page.

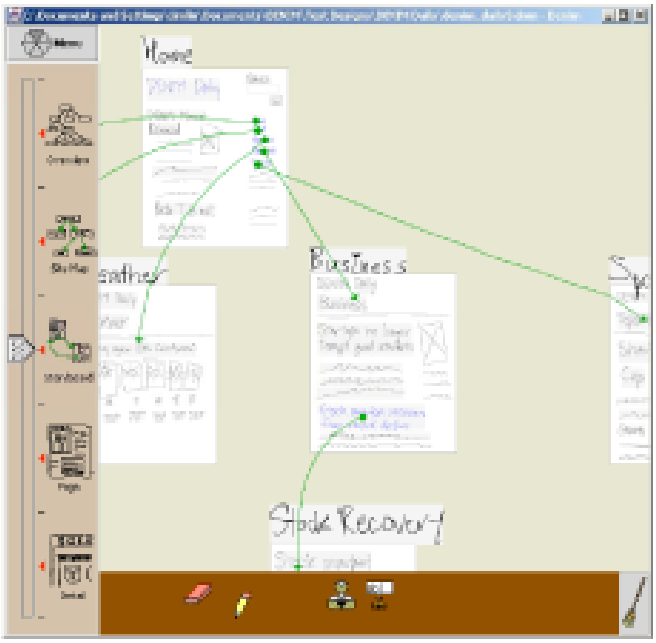


Fig. 2. DENIM [6].

There are two modes of DENIM: the creating mode in which the interface can be designed like explained above and the run mode. In the run mode, the links are clickable and the interface can be navigated on a screen like a website.

3.3.2 Why DENIM?

The visual language of DENIM [6] lets the designer take advantage of defining and reusing common interface elements like navigation bars. It is suitable to design large and complex systems. The zooming allows the designer to keep the overview of the whole system without loosing the details. The tool requires some more learning than PowerPoint but is still well adapted to the group of web designers that have less programming skills. With little learning effort, the tool can be used by the target group of web site designers, even if they have low programming skills [6]. In a newer version, DENIM supports HTML export of the prototypes.

The run mode allows usability testing at an early stage of design. By allowing more than simple click navigation, it is ready to help prototyping complex user interfaces for the web.

The disadvantage of DENIM might be that the visual appearance is rather poor as it is typical for low fidelity prototypes. But as the tool is designed for the early stages in the design process, this disadvantage can be neglected.

3.4 WARP

Web interfaces are used online, so why not use an online tool for prototyping? WARP (Web Application Rapid Prototyping) [1] is based on existing models and techniques that are brought together to support prototyping for web applications. Thus it is more an environment than a tool itself.

WARP provides an environment that supports the whole design process from the definition of requirements to actual coding of the software. This makes WARP The WARP environment is divided into two parts: the development environment and the execution environment *(see figure 3)*.

We will concentrate on the development environment first and within the development environment on the publishing editor. As it becomes visible in the figure, WARP integrates a lot of tools that can be used during the design process.

The pubishing editor (WPD) has to integrate the requirements that come from the database design and the content author with the ideas

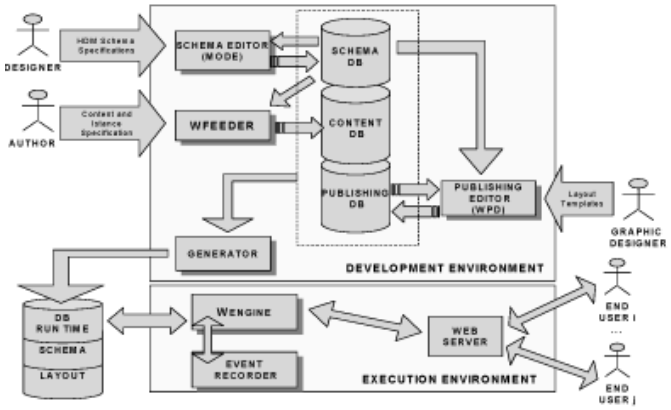


Fig. 3. WARP environment [1].

for visual representation of the content. This process is supported in WARP by XML based editing and by the use of external tools like DreamWeaver. [1]. As soon as the visual representation of the page is generated in a prototype, it can be refined for the final application. One advantage of WARP is that the visual interface of the prototype can be changed very quickly. This gives the design team the possibility to present alternative visual approaches without having to manipulate the back end of the prototype.

At the same time, WARP offers the possibility to work with different solutions of for access structures and hyperbase. The flexibility offered by WARP is combined with a workflow model that allows goal oriented prototyping.

The execution part will aid the design team to deliver the application on the web.

WARP appears to be a very complicated environment on first sight that relies on a strict method. It uses a lot of different tools which means that know how of using these tools has to be available in the design team. Designers will have to invest in training on several tools for prototyping and get used to the integrated method of WARP.

On the other hand the great advantage of WARP is that the work invested in the prototype will directly go into the final product. This is possible, because WARP supports flexible and fast prototyping in the early stages of the design process and at the same time supports coding and finalizing the web application. So WARP needs some extra work in training the design team but will pay back if the learned skills are used in many projects.

4 Classifications of tools for Prototyping of Web Interfaces

One way to make tools more accessible to designers is to reduce the complexity of choosing the right tool. This task can be managed by classifying the tools according to user needs. When we are talking about classifications of tools, we have the target to bring an order into the variety of existing tools. This section will give an overview of existing classifications of tools for prototyping web interfaces. The four tools presented in section three will be classified in the various schemes.

4.1 The Grade of Fidelity of the Output

For example, if we have the target to know what kind of output a tool can deliver, it will be interesting to know if it is a low-fidelity or a high-fidelity tool. This binary classification certainly cannot cover the variety of tools and has to be seen as two extremes on a continuum. This becomes obvious for example when scholars feel the need of introducing ”mid-fidelity” tools [3] like we have seen in section 3.2. If we use the definitions of Engelberg and Seffah [3] for low, mid and high fidelity protoyping, we can group the four presented tools from section three.

- Low-Fidelity tools deliver a rough sketch and are schematic and approximate. Their interactive functionality is very poor. They are suitable in the early design process for conceptualizing and envisioning the application.

- Mid-Fidelity tools deliver a more detailed and complete prototype but the objects are still schematic. The interactivity is simulated. Thus it can also be used for designing and evaluating interactive aspects of web interfaces.

- High-Fidelity tools deliver lifelike simulations with refined graphic design. They can be used as marketing tools or for prototypes where highly advanced interactivity and graphic design is needed.

Paper prototyping is a typical low fidelity prototyping method. Denim seems to be low fidelity at the first glance as well, but its features for interactivity (arrows) bring it towards the mid-fidelity tools. PowerPoint as described by Engelberg and Seffah can be used to generate mid-fidelity prototypes for web interfaces as well. The WARP environment can produce very sophisticated prototypes that are already on the way of a ready system. Thus it can be classified as a high-fidelity tool. If customers should work with the prototype or if the prototype is used for presentation purpose, one has to keep in mind that paying customers usually want to see very early how the final product will look like. Their expectation can only be met with high fidelity prototypes. This hypothesis stems from practical experience and not from scientific research and it would be interesting if it could be proved in an empirical study. By now I could not find any study that addressed this question in literature.

4.2 Features vs. Functionality - Horizontal and Vertical Prototypes

The purpose of a prototype can also be characterized by its "depth". Horizontal prototypes are used to sketch out the features of the web interface in general. They want to show the navigational structure and the main elements of an interface. If the intention is to go into the functionality of the web interface and model how it will react on specific manipulation, then a horizontal prototype is needed.

Paper prototyping can perform both ways of prototyping. The DENIM tool seems to be flexible enough for a horizontal and a vertical prototype as well. PowerPoint might not be so suitable for vertical prototypes and WARP can also cover both tasks.

4.3 Programming Skills that are Necessary to Use the Tool

As we have seen, some important actors of design processes do not necessarily have programming skills. If a project relies on these actors it will choose the tool for prototyping according to the skills needed. Moreover the learning process to use the tool can be a threshold for the acceptance of the tool in the design team.

Paper protoyping needs no programming skills at all but it needs the skill of sketching with pen and paper. PowerPoint is a standard tool that also needs no programming skills. The only threshold might be the implementation of links but this can be easily performed after some minutes learning. The DENIM tool relies on sketching but also need some knowledge about its specific visual language. The learning process might be quick but still can be a barrier. The WARP environment will be used by teams that are dominated by software designers rather than by visual designers.

4.4 Design or Implementation?

This distinction asks what a prototyping tool has to be able to do. Should it specialize on supporting the design process or should it cover the whole project cycle. There are good reasons for both approaches and it depends largely on the design team which approach will be appropriate. Tools that only support the design process leave more freedom to the design team [2] because any automatic implementation of a prototype will come to its limits when the web interfaces become very sophisticated.

Table 1. Classification Scheme

Classification	Paper Prototype	PowerPoint	DENIM	WARP
Fidelity	low	mid	low	high
Vertical Prototype	yes	no	yes	yes
Horizontal Protot.	yes	yes	yes	yes
Progr.Skills	none	none	mid	high
Focus on Design	yes	yes	yes	no
Export feature	no	no	yes	yes

Our four examples can be classified as well under this distinction. Paper prototyping and PowerPoint are just for Design. DENIM and WARP do offer export functions for implementation although the results still have to be refined [5][1].

Table 1 shows the four tools with the suggested classification criteria *(see table 1)*. There can be more classification schemes and we have seen that every scheme has its justification. Such tables might be a first step for supporting design teams in choosing their tools for prototyping and thus promote the use of prototyping in the design of web interfaces.

5 Conclusion

In this paper we have show the benefits of prototyping for web interfaces. Four exemplary tools have been explained to show the range of existing tools without neglecting that these four tools only give a small insight into the wide field of tools for prototyping web interfaces. In the fourth section the tools have been classified due to classification schemes that could be found in the literature.

Prototyping tools can support the design process of web interfaces. They are an investment but one with high return. Usability tests with prototypes can identify weaknesses in the early stage of the project. Prototypes can be a medium of communication within the design team and thus lead to rapid improvements of the system before programming has even started. In praxis, prototyping for web interfaces is still rarely used. One reason might be the vast amount of methods.

Technically there are a lot of good arguments to use prototyping in the development of web interfaces. As mentioned above, prototyping can improve the result of a project. On the other hand, it is very often seen as "extra work". Integrated environments like WARP or DENIM that rely on a consistent methodology might be very helpful for the design of web interfaces.

There is a lot of literature that promotes the use of prototyping but I could not find studies that examine the practical diffusion of prototyping. How many companies do use prototyping in their everyday software projects? And if they do not use prototyping tools, why? These questions have to be discussed and answered to improve the diffusion of prototyping techniques. Studies about the use of prototyping for web interfaces will have to open the perspective on the everyday (social) circumstances under which the development of web interfaces take place.

Although classifications might not always be taylor made for every single tool, they can be helpful for design teams that are searching for the right tool. In this paper only a few classification schemes could be chosen.

To create a useful tool for finding the right prototyping tool for the right task further research about existing classifications has to be made as well as a collection of existing tools for prototyping web interfaces. To ensure a comprehensive scheme of classifications the research might go further and evaluate the needs of web interface designers in an qualitative study.

References

[1] M. Bochicchio and N. Fiore. Warp: Web application rapid prototyping. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1670–1676, Nicosia, Cyprus New York, NY, USA, 2004. ACM.

[2] M. Bochicchio and R. Paiano. Prototyping web applications. In *Proceedings of the 2000 ACM symposium on Applied computing*, volume 2, pages 978–983, Como, Italy New York, NY, USA, 2000. ACM.

[3] D. Engelberg and A. Seffah. A framework for rapid Mid-Fidelity prototyping of web sites. Kluwer Academic Publishers, 2002.

[4] Y.-K. Lim, E. Stolterman, and J. Tenenberg. The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. *ACM Trans. Comput.-Hum. Interact.*, 15(2):1–27, 2008.

[5] J. Lin, M. W. Newman, J. I. Hong, and J. A. Landay. Denim: an informal tool for early stage web site design. pages 205–206, Seattle, Washington New York, NY, USA, 2001. ACM.

[6] J. Lin, M. Thomsen, and J. Landay. A visual language for sketching large and complex interactive designs. pages 307–314, Minneapolis, Minnesota, USA, 2002. ACM.

[7] P. Müller. *Little Boxes*. Books on Demand, Norderstedt, 2006.

[8] M. W. Newman and J. A. Landay. Sitemaps, storyboards, and specifications: a sketch of web site design practice. In *DIS '00: Proceedings of the 3rd conference on Designing interactive systems*, pages 263–274, New York, NY, USA, 2000. ACM.

[9] Nielsen Norman Group. Strategies to enhance the user experience. http://www.nngroup.com; last-checked: December 2009.

[10] R. Sefelin, M. Tscheligi, and V. Giller. Paper prototyping - what is it good for?: a comparison of paper- and computer-based low-fidelity prototyping. pages 778–779, Ft. Lauderdale, Florida, USA New York, NY, USA, 2003. ACM.

[11] C. Snyder. *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. Morgan Kaufmann Publishers, San Francisco, 2003.

[12] E. van Oost, S. Verhaeg, and N. Oudshoorn. From innovation community to community innovation. user-initiated innovation in wireless leiden. *Science, Technology and Human Values*, forthcoming, 2008.

[13] R. A. Virzi, J. L. Sokolov, and D. Karis. Usability problem identification using both low- and high-fidelity prototypes. Proceedings of the SIGCHI conference on Human factors in computing systems: common ground:236–243, 1996.

Usage of the Web for Various Prototyping Scenarios

Markus Zimmermann

Abstract—The emergence of new techniques for the web like JavaScript and AJAX and strong network technologies and servers are empowering software developers to create vivid web applications. In this paper we will analyze the benefits of web applications for one of the first steps in software development, the prototyping process. In combination with a vast amount of public accessible APIs, a web prototype can be simply mashed by combining different data and presentation sources, that are easily wired with code (those products are called a mashup). This paper will have a look at some projects, which employed this new prototyping approach by gathering the main findings of the corresponding studies: high fidelity prototypes through low effort. In addition, the used frameworks for rapidly creating web mashups will be figured out.

Index Terms—Web, Internet, Rapid, Prototyping, Mashup, API, AJAX

1 Introduction and Overview

At the same time Eric Knorr produced the term *Web 2.0* in late 2003 [20] because of the "genuinely new technology" in the web, he diagnosed that web services have reached a sufficient level to be an alternative to proprietary middleware (at least for new, not established projects). Now, six years later, we take a step forward and explore, how the web can be (and already has been) involved in the development process of new applications, especially the task of prototyping.

First, we will analyze the technological principles of the *Web 2.0* and examine the evolution that allows the development of web applications. This will lead us to *Patchwork Prototyping* by mashups. Second, we will apply ourselves to different frameworks, that are useful for rapid web prototyping without distinct knowledge of details. Third, we will emblaze different studies that attend to various web prototypes compared to traditional prototypes or existing products. In conclusion we will put all findings of web prototypes in context and lift web-prototyping out of different prototyping approaches.

2 Emergence of New Technologies for Rapid Web Development

Development of new techniques not only made it possible to display static hypertext sites, but also to dynamically create highly interactive *web*-applications, where client and server are closely interacting (a precondition for web-applications in general). As there are several technologies available, we will highlight the well-established techniques: *PHP* and *MySQL* as representatives of approaches on the server side. *DHTML*, *JavaScript* and *CSS* on top of all modern browsers, as well as helpers like *Libraries* and the new technology *AJAX* and different *Application Programming Interfaces (APIs)* .

2.1 Server-Side: PHP and MySQL

The scripting language *PHP* and the database engine *MySQL* are popular for creating web applications and often used in combination, so that most users mention them in one breath. Both Systems are robust, approved and work with good performance. So significant and high traffic websites like Wikipedia[28], YouTube[15] or Facebook[7] use these two technologies for their service.

PHP *(PHP: Hypertext Preprocessor)* was originally created as a toolkit for personal home pages, then continuously developed towards a free and general purpose scripting language. *PHP* can be embedded into a classic *HTML* page. The code is evaluated by the web server when the user invokes the website. [13]

- *Markus Zimmermann is studying Media Informatics at the University of Munich, Germany, E-mail: zimmermann@cip.ifi.lmu.de*

- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009/2010*

MySQL stands for *My Structured Query Language*, which is a free software for all major server platforms for accessing databases and tables on a web server. The user can put, modify and delete data in the tables and fetch the entries easily by requesting queries like `SELECT * FROM staff WHERE name = 'Smith'` (which will result in a list of all employees called "Smith"). MySQL has a comfortable interface to *PHP*. [22]

By combining the scripting language *PHP* with the database *MySQL*, it is possible to dynamically create actual or even live content out of a huge stock of data, where the contemporary appropriation of the same data as a static web document would have been impossible.

2.2 Client-Side: DHTML, Javascript, CSS and Libraries

While the server is responsible for creating and delivering content, the experience of interacting with an application comes up on the client-side: Data has to be rendered, formatted and conditionally modified. The user input has to be collected, evaluated and delivered to the server. Most browsers are standard-conform by now, so in the recent years it became possible to create and style applications for all browsers by joining the technologies (each in detail described by Christian Wenz [27]) *DHTML* for website modification, the browser scripting language *JavaScript* and the formatting-language *CSS*. *Libraries* make the usage and combination of all techniques easier.

DHTML stands for *Dynamic HTML* and allows the programmer to modify a website dynamically. While static websites stay changeless after loading them, *DHTML* (also known as *DOM-Scripting*) affects style and function of elements like images or paragraphs on the webpage when needed (for example by a *JavaScript* call).

JavaScript is a modern, object oriented scripting language executed in the web browser (with a java-like syntax). *JavaScript* enables the web designer to validate user input on the client side, to manipulate the website via *DHTML/DOM* and to compute, send and receive data.

CSS is a style-sheet markup language for appealing design of web pages and applications. As a powerful amendment of *HTML* or *XML* it can be used for formatting structured content (for example adjusting color, position, size or spacing of elements). Individual objects can be modified as well as it is possible to define central formats for classes of elements.

Libraries such as *Prototype* [25] (helper functions for *JavaScript* development) and the *Dojo toolkit* [9] (various user interface elements for *JavaScript*) provide a comfortable way for solving recurring tasks. For example requesting data from a server, *DHTML/DOM* modifications in the document or creating complex input elements like menus and calendars or drag-and-drop functionality.

2.3 Interaction: AJAX

Classic web interaction is limited to a hypertext structure: Whenever the displayed page or information is needed to be changed, this is accompanied by a reload of the whole page. This basic and slow interaction mechanism was the only way of web interaction for a long time. *JavaScript* improves the scope, but traditionally could not gather completely different data than delivered with the first page load.

In 1998, *Microsoft* developed a remote scripting component, enabling a web access interface for the *Outlook* email client (first as a java applet, then directly supported by their browser, the Internet-Explorer, as a *XMLHttpRequest*-Object). Microsoft's (and analog) approaches allowed the background loading of data without the need for refreshing the whole page. *Google* adopted this technology for the *Google Suggest* functionality: when you start typing `new york`, the part `new y` is enough for *Google* to autocomplete the possible search terms *New York City* or *New York Times*. All the loading is done in the background, without the page's reload.

The term *AJAX* was first mentioned by J. J. Garrett [10] in 2005: He called the technique of dynamical loading data in the background *Asynchronous JavaScript + XML*, "a fundamental shift in what is possible on the Web", featured by the mentioned technologies *HTML* (and its "dialect" *XML*), *DOM*, *JavaScript* and the *XMLHttpRequest*.

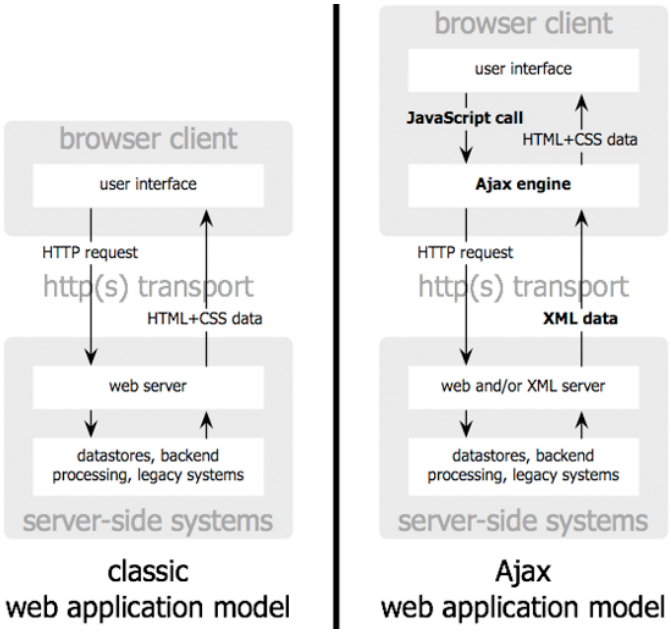


Fig. 1. Classic and *AJAX* web application model (original image in [10])

AJAX' mode of operation is described in *figure 1*: While a classic web application is just based on interaction between a browser (requesting pages on the client side) and a server (delivering the content gathered from various sources), the *AJAX* approach has an additional intermediary service level, the *ajax engine* (the simplest case is a single *XMLHttpRequest*), which requests (unformatted) content from the existing server infrastructure and attends to the styling and integration on client side.

We can contend that *AJAX* is rather a paradigm than a technology, powerful in combination with the well-established web techniques.

2.4 Application Programming Interfaces

An aspect of *Web 2.0* are the various and public available data and presentation sources from anywhere in the web. Content providers allow users to connect to their applications and retrieve information like database records, product informations or even graphical data like pictures, videos or maps. *ProgrammableWeb* [23], an API database, announces over 1.500 public accessible APIs by the end of 2009. The most popular services are (the numbers represent the APIs' usage based on 4506 analyzed mashups):

- *Google Maps API* [11] for geographical information and browsing maps (1867)
- *Flickr API* [29] for photo acquisition (488)
- *YouTube API* [12] to get any video content (429)
- *Amazon API* [1] for book information retrieval (327)

With the help of all the so far considered technologies and tools, combined with API service calls, it is possible to create entirely new web services by wiring the chunks and foreign informations, commonly known as *mashups*.

3 Utilities for creating Prototypes in the Web

Because of the simplicity of creating mashups, those can be used as rapid prototypes of future software products. The developer has no need to create dummy content close to reality and is able to reach a high fidelity prototype level without taking care of details.

We will distinguish between different integratable content: Foreign *data* (like metadata gathered from a Music Database), *applications/services* and *presentation* (e.g. display maps from Google) and look on different toolkits and frameworks valuable for a rapid prototyping process.

3.1 Mashups and Patchwork Prototyping

In 2007, Ingbert R. Floyd [8] did a research called *Web Mashups and Patchwork Prototyping* about novel practices in development with *Web 2.0*, open APIs and open source systems. He introductorily describes the two terms as following:

Mashups are websites combining data and services from multiple sources, all across the web, conceptually like a DJ creating a remix of multiple music sources. Mashups are being used for "rapid realization of creative ideas which would be too time consuming or expensive". A requirement for a mashup are public accessible APIs and data sources and the detailed knowledge about how to access them. This barrier can be lowered by the toolkits described in the following sections.

Patchwork Prototypes "use combinations of web services, mashups, locally developed code and open source software". They reduce development periods by rapid iteration cycles, incorporation of the prototype in the users' daily activities, and massive user feedback. The prototypes' innovations are user-driven.

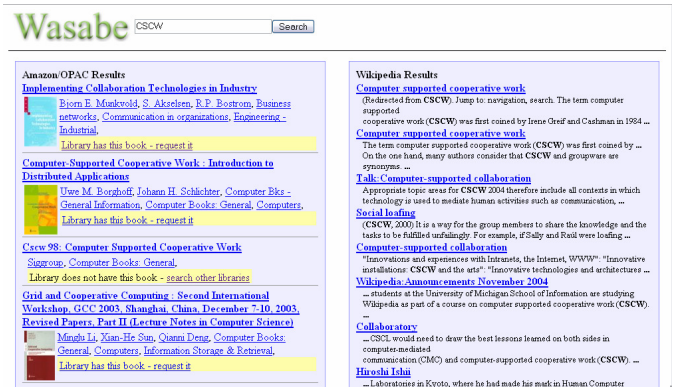


Fig. 2. Wikipedia-Amazon Search And Browse Environment, a simple mashup (original image in [8])

As an example for creating a mashup as a patchwork prototype, Floyd developed *Wasabe (see figure 2)* "as a prototype hybrid library

catalog system that allows users to search within a single interface both the detailed bibliographic information typically found in library catalogs as well as more general information about the topic of interest, typically found in encyclopedias" [8].

Wasabe used the *Amazon API*, the *Google Search API* and *Wikipedia* as additional data source. The first prototype of *Wasabe* has been created in less than ten minutes. The prototypes' successor is now present and used in the *A9* search engine.

He compared mashups to patchwork prototypes and found similarities: Technical innovations are user driven (that means better software through involved users). Although the methods are not new, Floyd finds out that technological innovations made patchwork prototyping affordable for single persons and small companies without hiring experts.

Nearly isochronal Cameron M. Jones [18] released his study *Patching Together Prototypes on the Web* with the same awareness: "Patchwork prototyping is a rapid prototyping approach" that saves speed and cost but also creates depth and high functional, high fidelity prototypes through mashups. By incorporation in the users' daily work, fast feedback can be gained while all the features and functionalities of the prototype are disposable.

3.2 Data Integration

A problem for a mashed-up web application or web patchwork prototype (in the following both are meant equally) is receiving data that is not reachable via API services or existent databases but only visible as a website or as the output of a present software. An experienced software developer would write some kind of converter. But in terms of rapid prototyping that is neither economic nor fast or universal enough.

The solution could be *programming by example*, Rattapoom Tuchindra [26] created "a Mashup building approach that combines most problem areas in Mashup building into a unified interactive framework that requires no widgets, and allows users with no programming background to easily create Mashups by example" called *Karma (see figure 3)*.

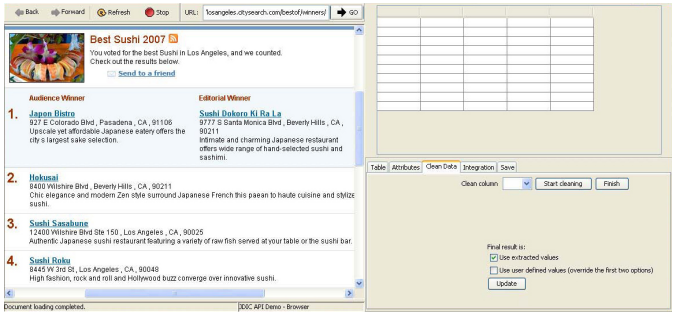


Fig. 3. The *Karma* interface: On the left an embedded browser, on the right a table for data extraction (original image in [26])

A prototype developer only has to browse a website with *Karma* and show the software where a data input (for example the search field) is located and how the resulting output is aligned (for example a table with corresponding names on the left and descriptions on the right).

The extracted data sets can be included directly into the prototype.

3.3 Application Integration

But even if API service calls are present, their invocation can be quite complicated. In his research, Björn Hartmann et al. introduced a tool called *d.mix* [14], which enables the prototype developer sampling those service calls through a proxy by demonstration (this approach is basically similar to Tuchindra's). The main difference is the accessibility of the content through API calls. In the background a mapping of *HTML* elements and their associated service calls takes place *(see figure 4)*. Users browse a website where API accessible content is highlighted by the proxy, mark elements they wish to copy, *d.mix* creates a corresponding service call and copies the results in its internal wiki. Thence the content and service call are hosted, can be modified and easily be integrated into a prototype.

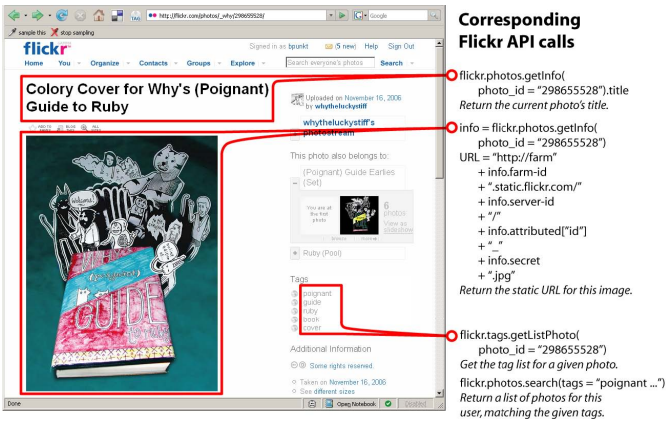


Fig. 4. Example of *d.mix* mappings between *HTML* elements and corresponding API service calls: Flickr Title, Image URL and Taglist retrieval (original image in [14])

3.4 Presentation Integration

As an alternative to *AJAX* (where *JavaScript* renders new content without reload), Jin Yu et al. created the presentation layer *OpenXUP* [30], an approach for creating highly interactive web user interfaces, consisting of a event driven thin client and server toolkit. It is based on the *Simple Object Access Protocol (SOAP)* and the *Extensible User Interface Protocol (XUP)*. *SOAP* is a technology for calling procedures on the server and exchanging data between client and server. *XUP* is a protocol used for communicating events and user interface updates on the web. The Framework enables the Model-View-Controller paradigm for web applications, offering a set of user interface components and bringing them closer to their desktop counterparts. Developers can create whole applications including user interfaces without assumed knowledge of many distinct web-technologies.

In his subsequent work, Yu proposed an *XML*-based presentation language *XPIL* for creating generic presentation objects without the need of interfaces and APIs [31]. The language is event based and the user interface components are easily applicable towards both desktop and web applications.

3.5 Prototyping Toolkit: WARP

Mario Bochicchio describes an environment called *Web Application Rapid Prototyping (WARP)*, that is suited for fast prototyping of web applications [3]. *WARP* supports the user in the complete prototype design process, from requirement analysis via design of the software's functionality to the complete coding and data handling process. The complete creationary process as well as the phase of execution is taking place in the web. Therefore *WARP* is covering a set of online tools:

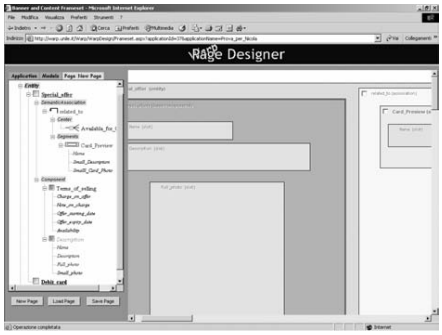


Fig. 5. The online tool *WARP* Page Designer (original image in [3])

- The *Schema Editor* for creating the application's model and exporting a *XML* scheme.
- The *Feeder* module that is complementary to a content management system. It can produce a sample of actual data.
- With the *Page Designer* the user can visually define the presentation of the Web Application *(see figure 5)*.
- A *Generator* that creates a runtime Database for caching purposes.
- The core *Engine* that dynamically delivers the pages based on the schemas, contents and layouts. By recording all the events, the usage of the prototype may be analyzed and evolved.

WARP, introduced in 2004 and presented as "first environment for fast-prototyping completely on-line", is an innovative but only partly finished project. Although it is apparently discontinued in its development process, it could be a trendsetting toolkit for all-in-one web-prototyping.

4 PROJECTS

Now that the technical fundamentals have been highlighted, we will discover some tangible projects, which had their origins in web prototypes. We will have a look at a *Crisis Management Prototype* and a *Healthcare Workers Service Prototype*, both powered by the *Google Maps API*. Furthermore we graze the different development steps of a *Mixed-Fidelity Prototype* of a task planner for Mars surface operations and we will end with considerations of *Clinical Information System Prototypes*.

4.1 Crisis Management Prototype: CHEOPS

CHEOPS 2.0 is a prototype of a replacement for the *CHEOPS* geopolitical risk and crisis management system designed in 1997, helping the military to understand and classify the situation.

In his paper, Francis Rousseaux [24] is describing the system and comparing the complexity of the creation of the original software to the creation of the prototype.

Table 1. Resources needed for CHEOPS and CHEOPS 2.0 (adapted from [24])

	CHEOPS		**CHEOPS 2.0 Prototype**	
	Software/ Equipment	Men/ Month	Software/ Equipment	Men/ Month
Maps/Images	20%	3%	0%	5%
Geographical Information System	70%	**70%**	0%	**0%**
Crisis Management System	2%	**20%**	0%	**65%**
User Interface	n/a	n/a	80%	20%
Other	8%	7%	20%	10%

CHEOPS had as components a commercial database management system, a commercial geographical information system and the core functionality. As we see in *table 1*, most of the development work and time (70%) was spent for the geographical information system, while the core system itself required only 20% of the attention.

The new prototype, *CHEOPS 2.0*, is a web application powered by the *Google Maps API* as geographical information system and *AJAX* techniques. *Table 1* also shows the effort taken for the single steps, while the geographical information system was now an external application, most of the time could be spent for the core functionality (65%).

Rousseaux's main findings are the increased productivity in prototyping but limitations for real applications on the other hand. The product could not have been validated, secured and maintained. This can not be guaranteed while using external components.

4.2 Healthcare Workers Service Prototype: GEOHEALTH

In his paper *GeoHealth, A Location-based Service for Nomadic Home Healthcare Workers*, Claus M. Christensen describes the implementation of a functional prototype, "which supports distributed and mobile collaboration through representation of live contextual information about clients, co-workers, current and scheduled work activities, and alarms adapted to the users' location" [4].

The *GeoHealth* web prototype heavily uses the *Google Maps API*, *Web 2.0* techniques and a *GPS* module for positioning. The prototype was developed after design sketching, paper prototyping and technology exploration *(see figure 6)*. The prototype was evaluated in a field study and was used for way finding assistance and task management for workers.

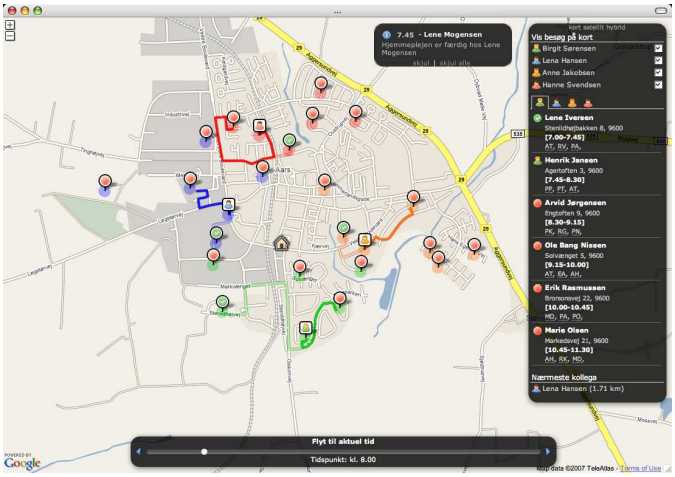


Fig. 6. The GeoHealth Home Screen (original image in [4])

While Christensens main findings deal with the improvement of the UI, he casually mentions the benefits of web prototyping: The web application resulted in a user friendly, high fidelity and real life prototype, with the advantage of mobility. But the future technological platform was not sure yet, a tangible application would follow.

4.3 Mixed-Fidelity Prototype: SPIFe

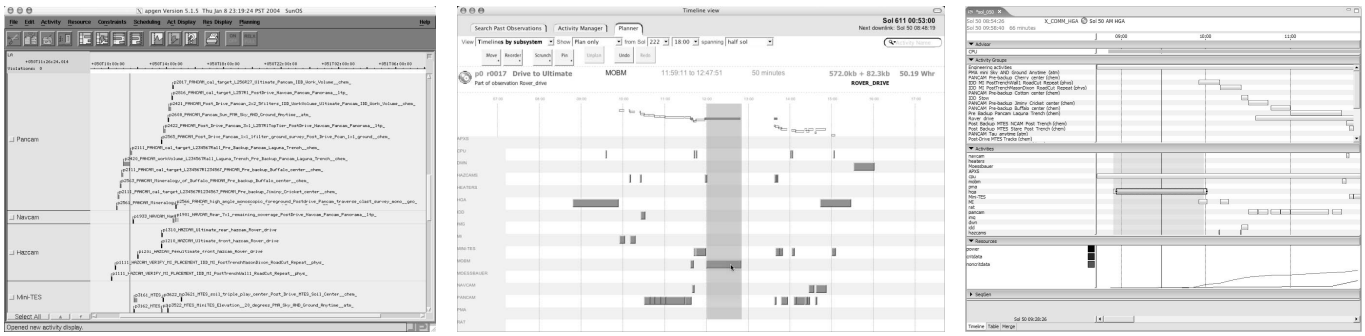
Michael McCurdy et al. [21] compared the renewed web prototype (*SPIFe*) of a preexisting tool (*MAPGEN* Activity Planning Tool) to a tool based on the eclipse platform that has been developed (*Ensemble*) using the prototype *(see figure 7)*.

The preliminary case: *MAPGEN*, an activity scheduling tool for the Mars Exploration Rover shown in *figure 7(a)* had to be redesigned because of missing functionalities and the lack of performance, after identifying the goals for the new product.

The web prototype *SPIFe* took advantage of *HTML*, *DOM* and *CSS*, dynamically created by a server-side program as shown in *figure 7(b)*. The prototype was created by a single developer in one month. Through creating a web application, it was possible to measure the accurate timing data (a mentioned goal was to increase the performance), what would not have been possible when using traditional user interface prototyping methods.

The final application *Ensemble* was built using the Java Eclipse Development Framework, which allowed to transport the gained results quickly into working software, necessarily being high fidelity. *Ensemble* can be found in *figure 7(c)*.

McCurdy et al. compared and tested all the three software artifacts, the results were (as expected) a slow *MAPGEN* tool. But for all tasks, *SPIFe* and *Ensemble* showed no significant differences in usability. Even though *Ensemble* was a high fidelity software product compared to *SPIFe* with its high fidelity data depth but low fidelity user interface. The prototype was as useable as the productive version (apart from the lack of graphical refinement).



(a) The *MAPGEN* Activity Planning Tool, used during the NASA Mars Exploration Rover missions in 2004

(b) The *SPIFe* Prototype Timeline

(c) The *Ensemble* Timeline

Fig. 7. The evolvement of an activity planning tool for Mars surface operations (original images in [21])

4.4 Clinical Information System Prototypes

An interesting sector for prototyping applications in the web are *Clinical Information Systems (CIS)*. Because of the complex and different procedures in clinical workday life, new ideas and processes have to be implemented quickly and tested regularly.

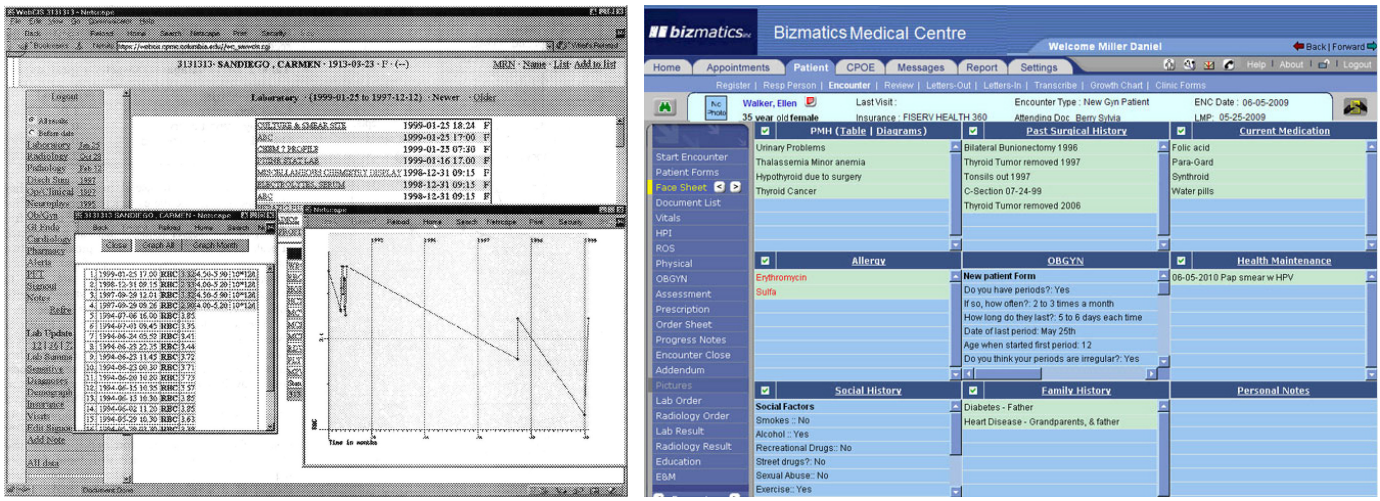
An old study of web development has been done 1995 by J. Cimino [6]. Subject is a Clinical Information System Prototype with the upcoming techniques *HTML* and *CGI* (an early approach for executing server sided scripts). Cimino used foreign internet based resources for his project, for example medical information retrieval, as an early form of a mashup. His findings at that time: The internet is an "exciting new environment" that excellently integrates with other resources.

Ciminos follow-up study in 1996 [5], was based on the earlier findings and put a closer look on web prototyping, again considering the CIS case. He still believes in the web as platform for rapid prototyping that makes "implementation of new functions quickly without large investments of effort", which had never been possible with traditional prototyping. He discovers the anyway available logfiles valuable for his user studies.

In 1998, G. Hripcsak built a web based CIS on top the existing *Columbia University CIS* [16]. The goal was to "make clinical information available to users wherever and whenever they need it", later potentially replacing the classic system. *Figure 8(a)* shows the main screen of the web application entirely written in *C* as a *CGI* application. Although the system can be distinguished as a prototype and Hripcsak mentions that the development cycle was fast, he discusses that all time-consuming "other aspects of the application life cycle" are still necessary. He criticizes that in the moment the application got finished, all used techniques (*CGI*, *HTML* and *JavaScript*) became obsolete and maybe unsupported by browsers. He concludes his study with the assumption, that "the best one can do is stick to a good clinical application architecture, with well-isolated modules, well-defined layers, and industry standard protocols."

A further disclosure and contemporary approach by J. H. Kim et al. [19] followed in 2001: Kim recognized the complex and ever-changing demands in CIS, too. Hence he developed of a rapid prototyping and clinical conversational system. Clinicians themselves (experts in acquaintance of clinical information systems but without programming skills and database knowledge) were now able to compose their own web-based clinical dialogues. Kims forecast was that applications will stay in the internet, because of the growing acceptance and the possibility to link between heterogeneous systems.

Todays developments are unambiguous: Because of the complexity and the demand for network compatibility, many CIS become web-based. For example the *IBA CIS* (developed in the early 1990s), a widespread system, has been redesigned because of lower maintenance cost and contemporary user interface look [17]. *BizMatics' PrognoCIS* shown in *figure 8(b)* is another example for the consequent port of existing systems into the web.



(a) Hripcsak's *WebCIS* prototype in 1998 (original image in [16])

(b) *Bizmatics Medical Centre*, a Web CIS Application in 2009 (original image in [2])

Fig. 8. Development of different Clinical Information Systems from a prototype to a running web application

5 Discussion and Conclusion

The use of the web for prototyping applications with the aid of *Web 2.0* and APIs is an ascent for prototyping, due to the development of programming possibilities and general conditions of the web. "Highly interactive" became an attribute for websites as well as for web applications. Prototyping scenarios benefit thereof, applications can easily be prototyped in the web with low effort, high fidelity prototypes can be created.

A limitation of the web is its stateless technology. Big effort has to be taken to (certainly) identify an user or to authenticate him. There is a huge offer of different browsers on several platforms (with slightly different techniques). Whenever a constriction to a single system is not possible, much work has to be invested for achieving a similar result on all browsers. A piece of software has to be validated, secured and maintained, that can not always be guaranteed when using foreign enclosed products.

But the benefits of web prototyping are predominant: In patchwork manner, the development cycle is fast. There is few need for knowledge of technical details and user interface development to achieve a wealth of functionality. The prototypes are highly interactive, here mid fidelity, there high fidelity, resulting in a mixed fidelity prototype. Prototypes (once adjusted) are platform- and device-independent, the worldwide access is guaranteed.

Future Work will probably be spend in even more simplifying the creation and data sampling process and imitating different target platforms (like embedded systems, special hardware or mobile phones). New upcoming techniques in the dynamic web will drastically blur the borders between classic applications and web applications.

References

[1] Amazon. Amazon web services. http://aws.amazon.com/, 2009. Last checked: 2009-12-09.

[2] Bizmatics. Emr software. http://www.bizmaticsinc.com/emr-software.html, 12 2009. Last checked: 2009-12-09.

[3] M. Bochicchio and N. Fiore. Warp: Web application rapid prototyping. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1670–1676, New York, NY, USA, 2004. ACM.

[4] C. M. Christensen, J. Kjeldskov, and K. K. Rasmussen. Geohealth: a location-based service for nomadic home healthcare workers. In *OZCHI '07: Proceedings of the 19th Australasian conference on Computer-Human Interaction*, pages 273–281, New York, NY, USA, 2007. ACM.

[5] J. J. Cimino and S. A. Socratous. Just tell me what you want!: the promise and perils of rapid prototyping with the world wide web. *Proc AMIA Annu Fall Symp*, pages 719–723, 1996.

[6] J. J. Cimino, S. A. Socratous, and P. D. Clayton. Internet as clinical information system: application development using the world wide web. *J Am Med Inform Assoc.*, pages 273–284, 1995.

[7] Facebook. Facebook platform news. http://developers.facebook.com/news.php, December 2009. Last checked: 2009-12-11.

[8] I. R. Floyd, M. C. Jones, D. Rathi, and M. B. Twidale. Web mash-ups and patchwork prototyping: User-driven technological innovation with web 2.0 and open source software. In *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, page 86, Washington, DC, USA, 2007. IEEE Computer Society.

[9] T. D. Foundation. The dojo toolkit. http://dojotoolkit.org/, 2010. Last checked: 2010-01-15.

[10] J. J. Garrett. Ajax: A new approach to web applications. http://www.adaptivepath.com/ideas/essays/archives/000385.php, February 2005. Last checked: 2009-11-20.

[11] Google. Google-maps api. http://code.google.com/apis/maps/, 2009. Last checked: 2009-12-09.

[12] Google. Youtube apis and tools. http://code.google.com/intl/de/apis/youtube/overview.html, 2009. Last checked: 2009-12-09.

[13] T. P. Group. Php: Hypertext preprocessor. http://www.php.net/, December 2009. Last checked: 2009-12-11.

[14] B. Hartmann, L. Wu, K. Collins, and S. R. Klemmer. Programming by a sample: rapidly creating web applications with d.mix. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 241–250, New York, NY, USA, 2007. ACM.

[15] T. Hoff. Youtube architecture. http://highscalability.com/youtube-architecture, December 2009. Last checked: 2009-12-11.

[16] G. Hripcsak, J. J. Cimino, and S. Sengupta. Webcis: large scale deployment of a web-based clinical information system. *Proc AMIA Symp.*, pages 804–808, 1999.

[17] IBAGroup. Web clinical information system. http://www.iba-it-group.com/en/case-studies/technology/other/a27c0d064f36a74f.html, 2009. Last checked: 2009-12-09.

[18] M. C. Jones, I. R. Floyd, and M. B. Twidale. Patching together prototypes on the web. 2006.

[19] J. H. Kim, R. Ferziger, H. B. Kawaloff, D. Z. Sands, C. Safran, and W. V. Slack. A web-based rapid prototyping and clinical conversational system that complements electronic patient record system. *Stud Health Technol Inform*, 84(Pt 1):628–632, 2001.

[20] E. Knorr. 2004 - the year of web services. In *CIO: Fast Forward 2010 - The Fate of I.T.*, volume 6, page 90. Gary J. Beach, December 2003.

[21] M. McCurdy, C. Connors, G. Pyrzak, B. Kanefsky, and A. Vera. Breaking the fidelity barrier: an examination of our current characterization of prototypes and an example of a mixed-fidelity success. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1233–1242, New York, NY, USA, 2006. ACM.

[22] S. Microsystems. Mysql - the world's most popular open source database. http://www.mysql.com/, December 2009. Last checked: 2009-12-11.

[23] ProgrammableWeb. Mashups, apis, and the web as platform. http://www.programmableweb.com/, December 2009. Last checked: 2009-12-11.

[24] F. Rousseaux and K. Lhoste. Rapid software prototyping using ajax and google map api. In *ACHI '09: Proceedings of the 2009 Second International Conferences on Advances in Computer-Human Interactions*, pages 317–323, Washington, DC, USA, 2009. IEEE Computer Society.

[25] P. C. Team. Prototype javascript framework. http://www.prototypejs.org/, 2007. Last checked: 2010-01-15.

[26] R. Tuchinda, P. Szekely, and C. A. Knoblock. Building mashups by example. In *IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 139–148, New York, NY, USA, 2008. ACM.

[27] C. Wenz. *JavaScript und AJAX*. Number 3-89842-859-1. Galileo Computing, 2008.

[28] Wikimedia and Various. Wikimedia servers - overall system architecture. http://meta.wikimedia.org/wiki/Wikimedia_servers#Overall_system_architecture, December 2009. Last checked: 2009-12-11.

[29] Yahoo. The app garden. http://www.flickr.com/services/api/, 2009. Last checked: 2009-12-09.

[30] J. Yu, B. Benatallah, F. Casati, and R. Saint-Paul. Openxup: an alternative approach to developing highly interactive web applications. In *ICWE '06: Proceedings of the 6th international conference on Web engineering*, pages 289–296, New York, NY, USA, 2006. ACM.

[31] J. Yu, B. Benatallah, R. Saint-Paul, F. Casati, F. Daniel, and M. Matera. A framework for rapid integration of presentation components. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 923–932, New York, NY, USA, 2007. ACM.

Patchwork Prototyping for Web Applications

Felix Heller

Abstract— Patchwork prototyping was first defined in 2007 and is a new approch to rapid prototyping. It is a participatory design concept which involves the prototypes' users by integrating a patchwork prototype into their daily work routine. The users' feedback is being extensively demanded to continously develop and improve the prototype. Therefore, patchwork prototyping is clearly a user-driven method. As patchwork prototyping is mainly a high fidelity prototyping method, it offers a relatively complex user interface which looks similar to a possible final product. To reduce the workload for developers, application programming interfaces, web services and open-source software are widely used and can be combined by using either simple programming techniques or by implementing more complex realizations. In this paper, the history of patchwork prototyping is examined and relevant terms are defined. Patchwork prototyping is compared to low fidelity, high fidelity, horizontal and vertical prototypes and prototypes using commerical off-the-shelf software. To conclude, the potential of patchwork prototypes is discussed and several project examples and development tools are presented.

Index Terms—Patchwork Prototypes, Rapid Prototypes, High Fidelity Prototypes, Participatory Design, Mashups, Open-Source Software, Application Programming Interface

1 Introduction

Only a few years ago in 2007, the term "Patchwork Prototyping" was invented by Floyd et al. [12]. It describes a new method of inventing a high fidelity prototype by patching together multiple components that can be used without further restrictions like open-source software. In addition to the already known types of prototyping, patchwork prototyping has some relevant advantages and allows to design a high fidelity prototype rapidly. In this paper, the facts about patchwork prototyping for web applications will be examined and will be compared to other methods of prototyping. To sum up, both limitations and strengths will be shown and a short overview over several patchwork prototype projects will be given.

2 History

Participatory design is a user driven-technique and in its early days, one of the first steps made was the initial computerization of specific work settings. Today, the conditions of work places have severly changed and are completly different to those of former days. The use of computers is an irreversible development which is widespread in most developed countries and the software that is used with these computers like word processor and spreadsheet programs is nearly omnipresent. Due to the ubiquity of the internet, web browsers are a standard software program nowadays and are widely used in workplace environments today [21].

For many years, the concept of free / libre open-source software (FLOSS) has been an ongoing trend in the software industry. A major advantage in comparison to commercially sold software is that a developer can liberally use components of a FLOSS program or even its whole source code for his own ambitions without considering license violations or having to pay fees to the original developers [21]. Furthermore, the innovative potential for open-source software (OSS) is named to be unlimited and innovation is present during desgin, development and even during use [11].

In summer 2005, the website HousingMaps.com was launched which fetches real-estate listings from the website Craigslist.org and filters the harvested data for usable geographic information. This collected data is then processed using the application programming interface (API) of Google Maps. At the end, the data is shown displayed on a Google Maps' map which is seamlessly integrated on HousingMaps.com. This was the first time when the term "mashup" was mentioned and when an API was used to create a new and innovative web application. Being able to use foreign resoures of the web for own purposes was a vital feature of this application [4].

Today, the "programmable web" is another trend which results in a movement from software installed and executed on a desktop computer towards web applications which have its data and program routines stored on remote servers. The applications can be accessed with web browsers so that a large installed base of those is a precondition [9]. So the internet is no more just a system for content storage and delivery but also an active platform for their users to empower their participation and innovation [13].

3 Definitions

In this section, the three terms mashup, rapid prototyping and patchwork prototyping will be defined to build a fundament for the more profound sections in this paper.

3.1 Mashup

A mashup integrates data sources and APIs into a single web application. The openness of data sources in the internet like XML feeds and documented programming interfaces encourages developers to invent own applications by mixing these sources [24]. If there is no open access granted by the manufacturer, developers have to use techniques like reverse engineering to gain access to the hidden program functions and data. This behaviour may also violate laws and result in criminal prosecution. In fact, creating an open web application's interface may also be an advantage for the manufacturer because an application may become more popular when users may access it through mashups as well. For example, the popularity of Google Maps is strongly founded on the open API which did not exist from the very first moment [13].

The enormous number of about 1 500 available APIs and 4 500 created mashups that are listed on ProgrammableWeb.com proves the high potential of this relatively new technology. The speed of development is still high - nearly 500 mashups were newly listed on ProgrammableWeb during the past six months [17]. While the number of existing mashups and APIs cannot be exactly disclosed, the correct values may be even higher [9].

To program a mashup, a specific level of knowledge is needed which depends on the complexity and scope of operation that the mashup shall have. If only HTML and JavaScript skills are required, undergraduate students without prior programming experience are able to learn how to program a mashup in only three weeks [13]. This short time can be realized as a consequence of the "black box" behaviour of most APIs which hides the complex functions and simply

- *Felix Heller is studying Media Informatics at the University of Munich, Germany, E-mail: felix.heller@web.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009*

returns the desired results to the developer. With the help of specialized online services like MapBuilder.net, even non-programmers with no experience may be able to create simple mashups. The process of "glueing" together the code is done by the web service [3]. For a detailled description of such services see *section 7*.

3.2 Rapid Prototyping

Rapid prototyping is a method to quickly develop and test new applications. When time is an important and precious resource, this is an advantage [13]. Using this method can help to improve the communication between developers and users and may return insights of possible strengths and weaknesses of the developers' ideas at a point in time before the whole application is finished. This can prevent spending money for disadvantageous developments [12]. Iteration is another relevant characteristic of rapid prototyping that allows to explore new features and alternatives beside existing solutions [11]. It supports creativity which is one of the main goals of rapid prototyping [12]. The traditional literature about software development judges the choice between a rapid and a high fidelity prototype as a fundamental tradeoff that has to be made in the design process of software programs [6].

3.3 Patchwork Prototyping

Patchwork prototyping is a relatively new approach to rapid prototyping and an example for a participatory design concept [10]. Furthermore, patchwork prototyping belongs to the class of cooperative prototyping methods [21].

One of the three main components of patchwork prototyping is that high fidelity prototypes can be rapidly iterated by using existing offline as well as online techniques like mashups, open-source software and web services [21]. The availability of many APIs and mashups *(as shown in section 3.1)* and high-quality FLOSS supports this fact [6]. Having an accessible source code is an important attribute of these components. For this reason, commercial off-the-shelf (COTS) software has major disadvantages and it is debatable whether it is adequate for patchwork prototypes [21]. This will be discussed extensively in *section 4.4*.

Third-party sources can be expanded with locally written code by the developers. To combine these parts, "glue" code has to be written to combine the single parts, although the required skill level for developers may be quite low regarding the fact that simple HTML hyperlinks already fit the requirements for patching together components [10]. Skilled experts though are able to produce patchwork prototypes that reach a considerably higher level of complexity than mashups. By all means, the time period consumed to produce a patchwork prototype is still short [6]. When building high complexity prototypes, time may be saved - for example - by using a framework [13].

The second main component is the integration of the patchwork prototypes into the end users' daily work activities. These prototypes are a "design in use" which can access and respond to these users' needs in a quite short period of time. For this purpose, the design and the implementation phases of the development process are blended. Moreover, the prototype can be gradually improved by replacing the integrated FLOSS components with production-scale modules [21].

The third main component is the the extensive collection of feedback so that patchwork prototyping can be clearly seen as a user-driven process *(see figure 1)* [11]. The already mentioned integration into users' daily activities also changes the conditions of the development process. The prototype is not defined in the first place, but it is expanded during the process of use, taking given feedback and practical experiences into account [10].

Floyd et al. [4] defined that developers and representatives of all kinds of user shall join a design team for patchwork prototyping. They divided the iteration process - which normally shouldn't take longer than a week - into the following five steps:

- Making an educated guess about the possible properties of the desired target system
- Searching and selecting tools that allow to implement at least a part of the designated functionality

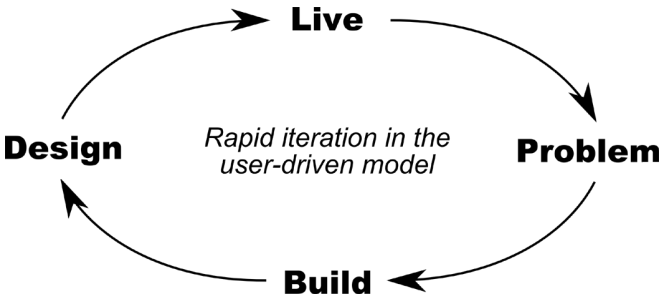


Fig. 1. Rapid iteration in the user driven model (adapted from Floyd, Jones, Rathi, Twidale [4])

- Combining the selected tools into a first draft
- Integrating the prototype into the end users' daily activities and collecting feedback from them
- Summing up the collected experience relying on the prototype building and the collected feedback and repeat the whole process

4 Comparison to Other Kind of Prototyping

Prototypes can be classified depending on the fidelity of the prototype and whether the focus is put on the functionality or on the feature richness. According to Tullis, the person who views a prototype classifies the fidelity of a prototype by the appearance of a prototype and not by the similarity of the prototype to the target application [20]. However, a neutrally precise differentiation is possible and so low and high fidelity prototypes as well as horizontal and vertical prototypes will be described and compared to patchwork prototypes. Furthermore, the use of commercial off-the-shelf software (COTS) in prototypes instead of open-source software will be compared and discussed. In *table 1* at the end of this chapter, some of the findings of this chapter about the strengths and weaknesses of patchwork prototyping, paper prototyping and prototyping with COTS are subsumed.

4.1 Low Fidelity Prototyping

Low fidelity prototypes can be developed and iterated with fast and cheap methods. Especially the paper prototyping method allows to construct prototypes with a short amount of time needed. To build a paper prototype, office materials like paper, pens and markers are sufficient enough which means that low fidelity prototypes are inexpensive *(see figure 2 for an example of a paper prototype)* [11]. This is quite similar to patchwork prototypes as they can be produced quickly and with a small amount of effort as well [21].

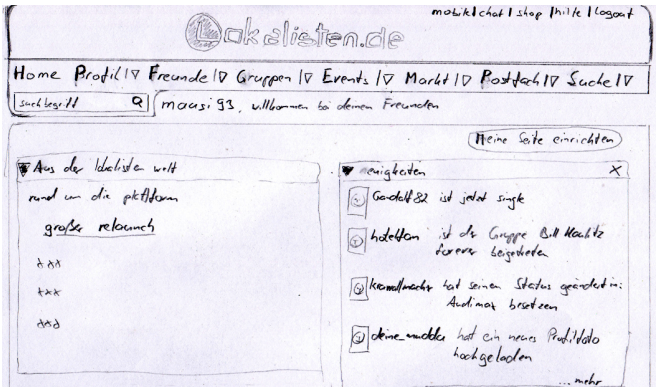


Fig. 2. Paper prototype for demonstrating and testing an improved design of the social community platform lokalisten.de (own work)

Beyond that, prototypes with low fidelity are a method of rapid prototyping like patchwork prototyping, too. On the contrary, they offer

only a limited functionality at most and shortened interactive elements which results in the limitation that low fidelity prototypes can show only a design direction with some details. For example, the content of a paper prototype can be simple and contain just a number of menus or static windows to match its designated use. Nevertheless, low fidelity prototypes can help to decide about fundemental design issues like the position of control elements. They are useful when it comes to the visualization of an interface. This qualifies low fidelity prototypes for the use in the early phase of gathering requirements and in user interface design teams. Using a lot of low fidelity prototypes is an effective instrument for identifying the requirements of both market and users [18]. By contrast, the tasks that users have to fulfill with low fidelity prototypes are completly artificial which may lead to prototypes that are designed for artificial use, but not for the real world [12]. In comparison to this, high fidelity prototypes like patchwork prototyping offer a relatively high complexity, are fully interactive and their design is closer to the one implemented in the final product [6]. On the other hand, patchwork prototypes are not effective for gathering requirements a priori [12].

A low fidelity prototype is often shown and demonstrated by a trained facilitator who explains the major functions and issues to the user. In order to simulate real functionality, these prototypes are accurately scripted. A major disadvantage in comparison to patchwork prototyping is that some of the features of a prototype have to be demonstrated to the user resulting in an unwanted influence on the user's opinion [18]. On the other hand, the facilitator may help the user instantly when facing problems so that feedback can be collected directly and a breach in the workflow can be prevented effectively. Comparing the fidelity of prototypes, Nielson arrived at the conclusion that many inconsistencies may remain hidden when using low fidelity prototypes. With high fidelity prototypes, the user is not influenced during the usage of the prototype by another person but cannot get immediate help if problems occur [14].

For the creation of low fidelity prototypes, no or only little knowledge of programming languages is required [18]. Compared to patchwork prototypes, low fidelity prototypes can be built with office materials and especially without a working computer system. In later phases of a project, this can result in new problems when programmers have difficulties to assimilate a low fidelity prototype template. The lack of detailed design specifications may produce an inappropriate design when the programmers have not enough experience in developing user interfaces on their own [18].

4.2 High Fidelity Prototyping

As patchwork prototyping belongs to the group of high fidelity prototypes, the following paragraphs name attributes of patchwork prototyping as well. Possible differences between standard high fidelity prototypes and patchwork prototypes will be discussed in detail.

High fidelity prototypes like patchwork prototypes "simulate real functionality" [11] and offer a complete interactive user interface. Instead of focussing on design and layout issues, high fidelity prototypes allow to get feedback how users navigate in the application [18]. In addition, it can be tested whether the design and user models match [22].

As a high fidelity prototype has almost the same behaviour as the final product, the users are confronted with the same kinds of error in the same places. This means that they can get a "feeling" of how the final product would work like and thus help to improve the behaviour of the prototype. Rudd describes a prototype in this case as a "test vehicle" [18]. Therefore high fidelity prototypes can be easily used without further restrictions as tools for exploration and testing as well as for marketing purposes. The already provided usability of the prototype allows the producer to promote it efficiently and therefore animate potential buyers.

As a result of the quite complete functionality of high fidelity prototypes, they may be used as a "living specification" [18] for both programmers and developers. Instead of making own considerations, they can simply start the prototype and examine how a specific problem has been solved. This can save not only time but also money when the developer is forced to make an own decision on a design issue which later on has to be withdrawn.

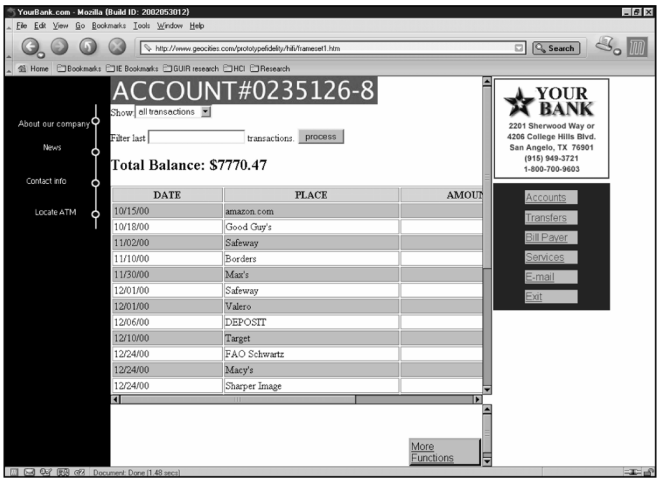


Fig. 3. High fidelity prototype of an account history page of a fictitious homebanking website *(original image in [23])*

Furthermore, the final program can be adapted to the users' needs in a better way. While feedback is already available after a short period of time when users have worked with the prototype, help utilities and documentation can be adapted better to their needs. Usually some time is needed to modify high fidelity prototypes, but patchwork prototypes can be modified quickly so that users who gave feedback can see that their desired improvements are quickly integrated. This may motivate them to use the prototype even more intensively and generate more feedback, resulting in an overall greater acceptance of a prototype. Since a high fidelity prototype can be used like the full featured final program, the usability can be evaluated and test cases can be constructed early in the software development process [18].

One of the disadvantages of high fidelity prototypes are its high development costs [18]. In comparison to them, patchwork prototypes try to bypass this problem by using third-party components like application programming interfaces and open-source software. "Outsourcing" the efforts can thus lower the own development efforts [6]. These measures also reduce the amount of time needed because of reducing the workload to the relatively simple "glue" source code [10]. Instead of taking several weeks to program the remarkable range of features included in a high fidelity prototype, patchwork prototypes can be developed in several days [4].

Compared to low fidelity and patchwork prototypes, high fidelity prototypes always require skilled developers who have to invest much of their knowledge into the developing process even though facilitations like high-level languages may support them [18]. As already mentioned, part of the concept of patchwork prototypes is to use much code from foreign sources to shrink the own development process. As a result, even untrained developers may be able to create patchwork prototypes by simply using HTML hyperlinks because the task of programming complex code is already done by the developers of the foreign code. [13] [10].

In conjunction with high development costs comes the argument of developers that a prototype is a waste of time and money and is seen as a unneccessary duplication of effort. Additionally, the argument may be raised that interface tests can also be done when it is already fully coded. Even if the developers can be convinced to develop prototypes, only few design approaches can be realized due to the fact that further approaches would overdraw the budget [18]. The savings of both money and time because of using patchwork prototypes may allow it to realize a prototype even in the case of a small budget and may open the possibility to build several more prototypes to prelude an effective design process. Still, the concepts of high fidelity and patchwork prototyping are not perfect for numerous conceptual approaches [18].

In the context of Web 2.0, the term "perpetual beta" [15] has been

introduced to describe the phenomena of software that is long-lastingly marked as a "beta" version which is constantly improved and updated. Popular services like Google Maps or Flickr have been in this phase for a long time [15]. A similar approach can be made towards high fidelity and patchwork prototypes: Due to the already included features in the unfinished product, customers may want to use the prototype as an already finished software product when it offers enough features and cannot be substituted with other available applications. A lack of understanding may arise when those customers do not understand that a prototype still lacks a well tested, documented and stable base of source code and intensive testing before it can be seen as a finished product [18].

4.3 Horizontal and Vertical Prototypes

While a fully functional and detailed program needs a lot of resources like working hours and money, a tradeoff has to be made to reduce the consumption of resources. This means that prototypes can be assigned to one of the two categories focussing on the scope of a prototype that Floyd [2] defined in 1984, its positioning depending on features and functionality can be seen in *figure 4*:

- Vertical prototypes have only several selected features included but they are realized almost completely like in the final product.
- Horizontal prototypes have a lot of features of the final product implemented but the features are only superficially complete.

When time is an issue, vertical prototypes allow to demonstrate at least the functionality of some features in a high fidelity prototype but does not cover the complete bandwidth of the final system. For this reason, low fidelity prototypes like paper prototyping are normally vertically integrated [18]. The depth of the features of vertical prototypes is "explored [...] through all layers" [12] and allows the developer to get detailed information about strengths and weaknesses of the implemented features [12]. The common ground of patchwork and vertical prototypes is that they include several features on a high level of functionality which is close to the final output of the product development process [18].

In contrast to vertical prototypes, horizontal prototypes cover nearly the whole breadth of a program resulting in a compulsive tradeoff to implement the functionality superficially [18]. In most cases, the layer in which the functionality is realized is the user interface which allows to analyze design issues and collect feedback for further decisions [11] [18]. Furthermore, users of a horizontal prototype can get an impression of how many different functions the complete program may contain without "plumbing" [12] the depths [12]. As with vertical prototypes, users and developers can receive a good impression of the width of a system with patchwork prototypes [10].

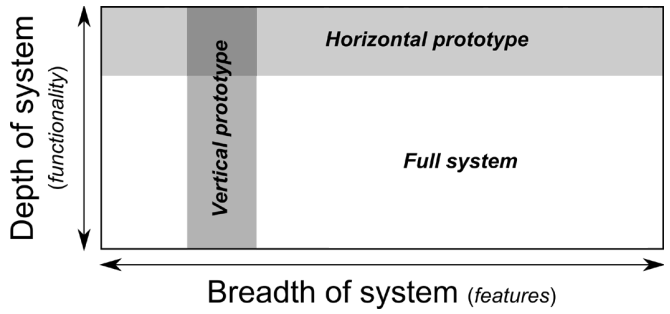


Fig. 4. Horizontal and vertical prototypes (adapted from Jones, Floyd and Twidale [11])

As shown before, patchwork prototypes share similarities with both horizontal and vertical prototypes. It can be concluded that patchwork prototypes thwart the categorization of Floyd [2] while other prototyping methods like paper prototyping can be assigned to one of the two categories without problems.

Due to this combination of attributes of horizontal and vertical prototypes, Jones, Floyd and Twidale [12] consider that the economics of developing may be problematical. High fidelity vertical prototypes with the breath of horizontal prototypes are much closer to a final system than vertical prototypes normally are. This means that an equivalent of the finished system has to be developed from scratch to fulfill the prerequisites of a both horizontal and vertical prototype. This can result in a lack of time and money. Furthermore, while horizontal prototypes span the breadth only within a specific level, both horizontal and vertical prototypes include a large functionality although several approaches to a satisfying design may be necessary. Nevertheless, patchwork prototyping allows to combine the two contradicting sorts of prototypes as a result of the methods of reducing the own efforts without lowering the feature richness as shown in *section 3.3*. This means that the method of stitching together different components guarantees the feature richness of a prototype. The developer is however still able to create a prototype that spans the width of a system like horizontal prototypes do [12].

4.4 Prototyping with Commercial Off-The-Shelf Software

Commercial off-the-shelf software (COTS) has some major differences compared to free / libre open-source software. In spite of free and editable source code, COTS products are closed systems in which own manipulations cannot be realized. To be able to use COTS, license fees have to be paid which can be due non-recurring or recurring. Furthermore, payments for maintenance may have to be made. While COTS is developed by a market participant, the buying of COTS can result in a lock-in effect [19] and in dependence on the vendor. These facts can be clearly stated as severe disadvantages in comparison to FLOSS.

As a market participant, the further development and updating cycle is mainly influenced by the needs of the market and not by the own needs. This is similar to popular open-source software although the vendor of commercial software can dictate the time period during which support and updates for a version branch are granted without having to respect the opinions in a strong community that contributes to the product.

Furthermore, a developer has no control over the functionality and the performance of COTS as it behaves like a black box which can be notably complex. This results in a handicap for the development process as the requirements for a program do not define the capabalities but just the other way around. Not implemented features in COTS cannot be added simply by modifying the source code because it is not accessible. If a closed product has a helpful interface that offers the desired functionality (like an API), this difficulty is not relevant.

Another disadvantage of COTS is the fact that the promises about the features of COTS in advertising campaigns may deviate from the real functionality of the end product. The black box character of COTS may aggravate this issue even more because it complicates the process of verifying the software's reliability. This problem can be addressed by the producer by complying to relevant open commercial standards and by publishing a roadmap of further software releases.

A major issue for prototyping and patchwork prototyping in particular is the possibility to combine several different techniqual components to one seamless application. With COTS, an additional challenge is the lack of programming interfaces for exchanging data with other software as well as the architectural mistake that makes COTS programs act like "sole rulers". This means that interoperability of commercial off-the-shelf software can be a serious issue when patching it together with other software components.

A positive aspect about some commercial off-the-shelf software products is that they are widely used and therefore can be seen as a stable and advanced technology product. Due to the existence of the commercial production process, expert support directly from the developers may also be available although it may not be free of charge. Like open-source software, commercial software used in a prototype can help to lower the own development efforts [1].

When analyzing the existing literature on patchwork prototyping, two different opinions about the use of commercial off-the-shelf soft-

Table 1. Comparison of patchwork prototyping with paper prototyping and prototyping with COTS (adepted from [12] and combined with the conclusions made in *section 4*)

Attribute	Patchwork Prototyping	Paper Prototyping	COTS Prototyping
Speed	++	+	-
Monetary costs	++	+	- -
Need for materials	++	++	o
Functionality	- -	+	+
Accessibility	++	o	-
Interface	o	+	+
Flexibility	+	++	-
Disposability	++	+	-
User attachment	-	+	++

Rating	Meaning
++ / +	strongly positive / positive
o	neutral
- - / -	strongly negative / negative

ware in patchwork prototyping can be found: In 2008, Twidale and Floyd [21] neglected the possibility of using COTS in combination with patchwork prototyping while in 2007, Jones, Floyd and Twidale [11] [12] were of the opinion that patchwork prototyping does not necessarily require open-source software. Furthermore, in [4] is stated that APIs of closed source code proved numerously that they can be implemented in various software applications. In their opinion, adapting methods could be used to implement COTS even into a working prototype [11] [12]. However, they saw the availability of production-scale FLOSS as a major fact for the emergence of patchwork prototyping [11]. It may be assumed that the change of mind of Michael B. Twidale and Ingbert Floyd is an outcome of further research that underlines the necessity of the use of software with modifyable source code.

Nevertheless, COTS has some significant disadvantages compared to open-source software like the limitations in patching together different components. These are caused by application programming interfaces of COTS which may be artificially reduced by the software developer. Furthermore, it is not guaranteed that an API is allocated at all. The closed source code precludes to alter the functionality of the software and denies the possibility to receive an impression of the underlying algorithms. To sum up, qualitative open-source software is the first choice for patchworking prototypes as it has none of the disadvantages of commercial software and the slightly different advantages can be disregarded as they are not serious [12].

5 Potential of Patchwork Prototypes

The method of patchwork prototyping has been compared to several other approaches in the previous *section 4*. While some advantages and disadvantages were already brought up, this chapter will explore them in detail and discuss possible causes and impacts.

5.1 Limitations

The development of high fidelity prototypes normally takes several weeks. In cases where decisions have to be made really quickly, this may take too long. As to speak of matters of time, patchwork prototyping is dependent on the staying power of the users as it is expected that they commit to a prototype on a long-term base. Thus, the users have to be motivated and convinced to use a patchwork prototype in their daily work activities and to commit feedback on a regular basis [12]. For this purpose, a motivated facilitator has to be appointed to actively stimulate the users and to collect feedback about the users' experiences [10]. Furthermore, significant leadership is required to ensure fast iteration cycles [21]. An unmotivated user base and insufficient feedback can otherwise lead to the outcome that the iterative development process collapses and the whole approach of using a patchwork prototype ends up in failure [10].

The usage of free / libre open-source software can result in a vulnerability of the own patchwork prototype as security flaws of included FLOSS programs may still be exploitable in the prototype [10]. Strong evidence for this risk are the events of project II *(see section 6.1)* where a patchwork prototype used the widespread internet community software phpBB. Due to a vulnerability of phpBB which was successfully used by an internet worm, the prototype was severly affected during the project phase [12]. Patchwork prototypes for web applications are in a higher danger because they can be accessed over the internet and are not deployed in a local and secure area.

Although seams of the process of patching together different components may still be recognizable in the prototype, users tend to see patchwork prototypes as already finished products. The existence of such "stitchings" seems to be of no advantage to underline the character as a prototype [10]. The main reason for this behaviour is the high functionality of a prototype while prototype users may not be able to understand the explicit differences between a patchwork prototype and a final product [12] [18]. With a further "socializing" of the prototypes' users, this effect may be eventually lowered. Incoherent design and usability may confuse users and developers of patchwork prototypes. They should try to hide the seams and the inconsistencies of different components [10] [12]. According to Jones, Floyd and Twidale, patchwork prototypes are not "renowned" for their good usability [12].

Furthermore, in some cases users may need different views of a prototype's features when using miscellaneous ways to access the application. This can go to such lengths that differences and seams between the different components of a patchwork prototype need to be completely imperceptible [10].

Another aspect of possible limitations due to the the users' behaviour may be that a used technology is disliked because the users were confronted with a similar unsatisfactory technology before. As single components of patchwork prototypes can be easily substituted, this issue can be addressed with little effort. The situation itself should be identified correctly before further steps can be taken [5].

In spite of the fact that patchwork prototypes can be realized by using programming or markup languages such as HTML which are quite easy to learn, patchwork prototypes that offer a higher level of functionality and complexity need to be developed by experienced personnel [13] [10]. This means that an effective iterative development process can only be guaranteed if developers do not have to spend a lot of time on reading source code and learning development environments. A lower implementation speed could be the consequence and could endanger the rapid development cycle [12]. For developing such complex patchwork prototypes, the tasks that developers have to perform are a limitation: Hard mental working has to be done when the used components are written in different programming languages and offer several different levels of abstraction [9].

This leads to another limitation of patchwork prototyping: Although it is cheaper than using commercial off-the-shelf software for prototyping issues *(see section 4.4)* and needs less time compared to high fidelity prototypes *(see section 4.2)*, a relevant amount of money has to be invested for maintaining the computer infrastructure and for paying the developers [21].

Unlike open-source software whose source code is entirely open, application programming interfaces offer only a predefined interface which may not span the entire breadth of an application. The proceedings inside remote APIs cannot be disclosed furthermore and are unknowable [9]. Into the bargain, a lot of application programming interfaces do not rely on a common standard, resulting in various conflicting models and data formats. A global authoritative control could define such standards, but there is no such institution at the moment [8] [9]. An exemplary exception is Google's OpenSocial API which is widely used by social network platforms [16]. Additional efforts may however be needed to transform data from one format into another. The single components potentially were not designed to cooperate with each other. "Custom marshalling" [9] thus could become a need. Other side effects may occur when components do not work as expected, for example when a data source is temporarily unavail-

able. Trusting the realiability of third parties leads to these limitations and limits the power of patchwork prototyping [9]. Despite of the high number of freely available code, there may be scenarios where the offered APIs do not fit and new solutions have to be created with possibly high efforts.

5.2 Advantages

The patchwork prototyping approach offers several strong advantages like the rapid iterative development cycles of patchwork prototypes which can be named as the "holy grail" of the high fidelity prototyping method. This rapidness leads to an overall improved design quality [4]. Additionaly, a strong advantage of patchwork prototyping is the complete functionality realized by patching several different components together which allows a fully interactive user interface that can be used well for both exploring and testing scenarios. Users of such prototypes probably feel like they are using a final product [18]. To sum up, the approach of patchwork prototyping uses a lot of the advantages offered by integrating qualitatively high FLOSS, APIs and web services [4].

In addition to that, patchwork prototyping demands developers as well as users a remarkable level of informality and flexibility [6]. This brings forth the possibility to easily add, substitute or remove components in a fast and straightforward way. New features can be integrated with little effort and the whole prototype can be set up within a few days or - at most - within a week [10]. Disparate information from different sources can be collected and transformed into a unique output, enhancing both usefulness and relevancy [16]. In special cases, even users may be able to contribute to the development process and improve the prototype on their own. Furthermore, patchwork prototyping is compatible with community-driven initiatives [4]. For these reasons, patchwork prototyping can be seen as a user-driven model and has many advantageous characteristics to support this [18].

As already mentioned in *section 3.3*, the collection of feedback is one of the main characteristics of patchwork prototyping. This implicates that the patchwork prototyping method allows to collect users' feedback to the full extent. Due to the software architecture (described in the first paragraph of this section), developers of patchwork prototypes can respond to users' desires effectively so that response times can be kept short. This results in the advantage that users receive the impression of being an integral part in the whole process [4].

In addition, patchwork prototyping enables the users to not only give general feedback but to state their opinions in a more detailed way. This makes the feedback even more valuable [12]. In patchwork prototyping enviroments, users are able to criticise specific instances because many different ideas and concepts can be demonstrated to them. Thus the users can get a certain understanding [4]. The low equivocality of the design space can be seen as a supporting fact in this matter [10]. To this end, the given feedback relates to a prototype which does not require to change the users' work enviroments gravely in order to adapt the prototype and which is not set up in artificial surroundings but can be used like in the real world [12].

Using the widespread technology of web browsers as runtime environment, patchwork prototypes become mostly platform-independent as web browsers are available for all major operating systems (Windows, Linux, Mac). Developers have the choice whether to use only simple markup languages like HTML or to write own source code in order to patch the different components together. This freedom of decision is another advantage of prototyping. It influences the amount of time needed to build a prototype as the use of simple markup languages results in a shallower and thus faster implementation [10].

If a patchwork prototype is successful and shall be finished to a final product, the transformation to production-scale components can be done piecewise when using a stable and modular framework. The advantage is that the prototype can be continously used without interruptions as well as the transformation can be carried out transparently for the users [10].

To conclude, patchwork prototyping combines a lot of advantages and can be seen as a unification of some attributes of low and high fidelity prototypes as well as of vertical and horizontal prototypes. These prototypes were already compared to each other in *section 4*. The advantages of low fidelity prototypes such as high speed implementation, (very) low costs and easily available materials are also part of the characteristics of patchwork prototyping as well as the fully interactive, functional interface in the user-driven process of high fidelity prototypes. Additionally, patchwork prototyping combines the two opposed categories of scope as it melts the breadth of vertical prototypes with the depth and high fidelity of horizontal prototypes [10].

6 Project Examples

In this chapter, two different web applications that use patchworking prototyping will be described and discussed.

6.1 Project I: Patching Together Community Tools

This project started in 1997 with the target to support processes of inquiry by constructing and evolving adequate web-based tools. As the project has been running for a lot of years, a long-term perspective on the design process can be assumed. Over the years, several different prototyes have been developed [12].

Six years later, in 2003, a first approach towards the use of third-party tools was made. These tools are customized for the use in communities. The project is furthermore completly based on free open-source software. The server environment is a typical LAMP configuration, including the use of Linux (operating system), Apache (web server), MySQL (database) and PHP (programming language). The prototypes can be accessed by anybody and their source code is published under a Creative Commons license. They already empowered widely heterogenous types of communities all over the world. On the one hand, this demonstrates the flexibility of the prototypes to be used in various kind of situation, but on the other hand, it disables the process of customizing the prototypes for a more specific use [12].

Due to a lack of knowledge and know-how, the prototypes in this project did not make use FLOSS technology at first. Hence, a lot of efforts were made to construct and develop the aimed functionality from the scratch. Later on, free web-based technology products like TinyMCE as a WYSIWYG editor and phpBB as a bulletin board were implemented and included into the prototypes. As already mentioned in *section 5.1*, the implementation of phpBB resulted in security vulnerabilities that were successfully exploited by an internet worm and resulted in damage to the prototype as a whole. Nevertheless, the usage of phpBB as a bulletin board is interesting when focussing on the aspects of patchwork prototyping. At the beginning, an instance of the phpBB bulletin board had to be manually installed and configured for usage by a specific community. The integration of the bulletin board was simply realized by putting a HTML hyperlink on he community's home page. The installation procedure was afterwards improved so that no direct interaction of a developer was required any more. An automatic procedure substituted the manual interaction and installed a new instance of the bulletin board. Until then, two different login possibilites were needed. To eliminate the rather redundant second login, the authentification and user management were thus melted together with the rest of the community home page [12].

As most users needed only the basic features of the bulletin board like posting replies and dividing their postings into different sections, the phpBB software was seen as overscaled and thus replaced by a "homemade" bulletin board with less features. The "homemade" component offered a better integration with the rest of the community platform. This concludes that the choice of an open-source software that fits the own needs is important to avoid further efforts needed to adapt applications that do not offer the right amount of functionality [12].

6.2 Project II: Combining Powerful Search Engines

The next example of patchwork prototyping is an approach to merge the output of different search engines wisely. It is called "Wasabe", an acronym for "Wikipedia-Amazon Search And Browse Environment", and is a hybrid search engine system *(see figure 5 for a screenshot)*. Wasabe's main feature is the search for both encyclopedic and detailed bibliographic information in one single step. Thus, the application programming interfaces of two major internet companies, by name

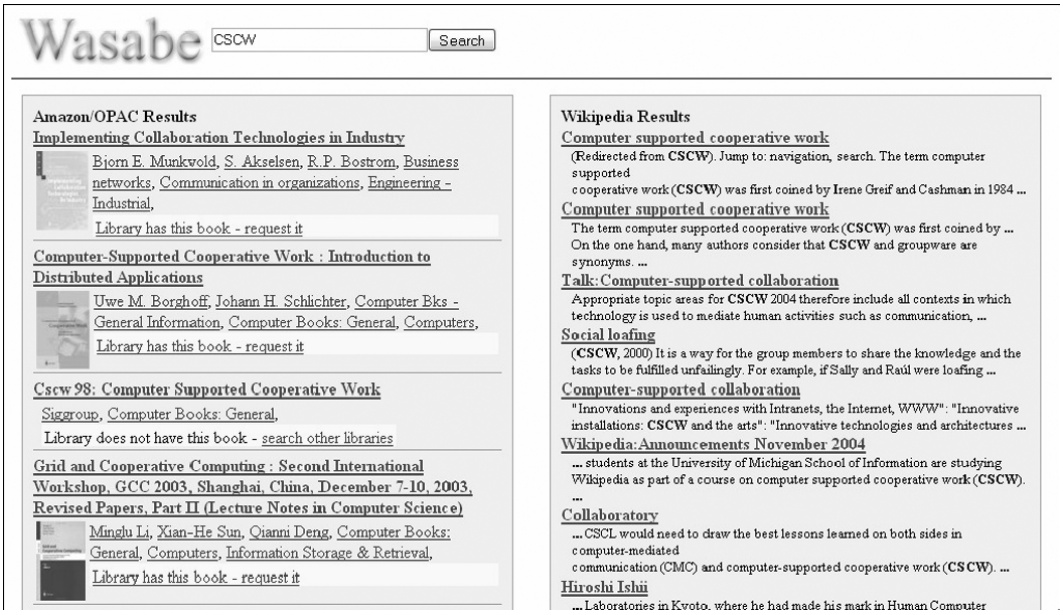


Fig. 5. Wasabe: A hybrid search engine system *(original image in [13])*

Amazon and Google, were used [10]. The Google SOAP API was thus used to search the encyclopedia wikipedia for information as it can be assumed that the wikipedia itself apparently did not provide an applicapble programming interface [4].

Three differently advanced prototypes of Wasabe were produced. The first prototype combined search results from Amazon as well as from the wikipedia (with the help of the Google API). The second prototype widened the search in the Amazon catalog and respected only the standard search results for the search term but also the information about related items. As the calculation of related items is done by an Amazon algorithm that evaluates the actions of Amazon users on their website, the second prototype of Wasabe adapted effectively the knowledge of the large Amazon user community for its own purposes. The second prototype furthermore copied the ISBN numbers of the Amazon search hits and pasted it into a query for a library web catalog to gather information about the availabilty of goods in a library. Due to the low response speed of the queried library web catalog, the technology of querying was altered to speed up the prototype [10]. Asynchronous JavaScript and XML (AJAX) [4] was implemented to change the method of querying the library catalog from server-side to client-side [10].

Unfortunately, the data about the exact size of the source code of Wasabe differs between a few dozens and multiple hundred lines depending on the literary sources [10] [4]. Anyway, the amount of source code needed to realize the three differently advanced prototypes is low and the first working prototype has been produced extraordinary fast in only ten minutes [4].

As a whole, all of the different Wasabe prototypes have remarkable strengths of the patchwork prototyping approach in common: The speed with which a working prototype can be created is extremly high and can - at least in this case - compete even with low fidelity prototyping methods like paper prototyping *(see section 4.1)*. In conjunction with this, the efforts needed to both build and improve a patchwork prototype can be really low without having to accept compromises regarding functionality and usability. Open access to large foreign data sources is furthermore a key component to get the prototype working. The advantage of harvesting in titanic databases of real data material supersedes the need for dummy data and demonstrates the functionality and helpfulness of an application in an even better way [4].

7 Development Tools for Web Applications

As mashup programming requires skills in developing web applications, a new class of tools has emerged to allow less trained developers and even people with no programming experience to put together different input sources on the internet to one common output [24] [9]. These applications are named as mashup development environments (MDE) [9] and describe web-based visual programming languages [8]. Although these applications are primarily focussed on the production of mashups at the moment, a closer look can be risked as the rapidly ongoing evolution of web-based applications may lead to tools with which more complex tasks like the development of patchwork prototypes could also be fulfilled.

Mashup development environments like Yahoo! Pipes visualize the dataflow between different data sources and allow to transform and combine the collected information. At the end, the processed data flows into one main output. To realize this, Yahoo! Pipes offers several different programming interactions. On the one hand, this simplifies some operations, but on the other hand, some may get even more difficult [9]. Advantages as well as disadvantages can be seen in the area of error handling as well. As the use of Yahoo! Pipes may mitigate a lot of different error types, new kinds of error may occur and the source of errors may be obfuscated [8].

Another slightly varying approach is d.mix which can be used to rapidly create new web applications. A difference to systems like Yahoo! Pipes is d.mix' attribute of providing a "site-to-service map" [7] through a programmable proxy system. This is already a more advanced approach as it allows to already edit the source code which collects and transforms the harvested data. The collection of such data for the use in d.mix is quite easy as new data sources can be simply defined when browsing a web site - preconditioned that a special template has been created to define how the data has to be processed for the use in d.mix. The attributes of the collected data can be altered using property sheets in the d.mix application. This enables even non skilled users to put together their own simple application and advanced developers are able to modify the code (written in Ruby) which processes the data. Furthermore, if data from a web site shall be collected and no application programming interface is offered, d.mix can be used to parse data directly from the HTML source code. The access through APIs is normally accelerated, but fetching normal HTML web sites may slow down the whole application [7].

To conclude, mashup development environments may already share some similarities with web-based tools which could produce patch-

work prototypes. Even though there is no possibility yet to include source code of FLOSS or even own source code, these tools may be a first prospect to more complex development tools in the future.

8 Conclusion

To sum up, patchwork prototyping is a promising new methodology which consequently advances the possibilites of existing kinds of prototyping by combining some of the main advantages of these prototyping approaches with both solid and rapidly evolving kind of software. While the number of mashups and application programming interfaces has grown rapidly over the last years, the field of patchwork prototyping may play an important role when it comes to future-proof prototyping methods. However, one should not lose sight of the possible negative aspects of patchwork prototyping as slackness may clear some of the advantages.

The unequal opinions in several papers about the use of commercial-off-the-shelf software in patchwork prototypes underlines the need for further research about this topic. As it can be seen in the list of references at the end of this paper, only few authors have examined patchwork prototypes by now so it can be stated that the "breakthrough" for this technologyhas has not yet happened. At the end, as internet technologies are emerging really fast and technologies like mashups and APIs are already widespread, this might be only a question of time.

References

[1] B. Boehm and C. Abts. Cots integration: Plug and pray? *Computer*, 32(1):135–138, 1999.

[2] C. Floyd. A systematic look at prototyping. *Approaches to prototyping*, pages 1–18, 1984.

[3] I. R. Floyd. Using mash-ups for end-user rapid and responsive prototyping in collaborative environments. *Communications of the ACM*, 37(4):21–27, 2006.

[4] I. R. Floyd, M. C. Jones, D. Rathi, and M. B. Twidale. Web mash-ups and patchwork prototyping: User-driven technological innovation with web 2.0 and open source software. In *HICSS*, page 86. IEEE Computer Society, 2007.

[5] I. R. Floyd and M. B. Twidale. Issues in infrastructure development: Proposed interventions for addressing third order issues. In *CSCW*, pages 1–4, 2008.

[6] I. R. Floyd and M. B. Twidale. Learning design from emergent co-design: Observed practices and future directions. In *Proceedings PDC*, pages 1–4, 2008.

[7] B. Hartmann, L. Wu, K. Collins, and S. R. Klemmer. Programming by a sample: rapidly creating web applications with d.mix. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 241–250, New York, NY, USA, 2007. ACM.

[8] M. C. Jones and E. F. Churchill. Conversations in developer communities: a preliminary analysis of the yahoo! pipes community. In *Communities and Technologies '09: Proceedings of the fourth international conference on Communities and technologies*, pages 195–204, New York, NY, USA, 2009. ACM.

[9] M. C. Jones, E. F. Churchill, and M. B. Twidale. Mashing up visual languages and web mash-ups. In *VLHCC '08: Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 143–146, Washington, DC, USA, 2008. IEEE Computer Society.

[10] M. C. Jones, I. R. Floyd, and M. B. Twidale. Patching together prototypes on the web. *CSCW*, pages 1–4, 2006.

[11] M. C. Jones, I. R. Floyd, and M. B. Twidale. Patchwork prototyping: A rapid prototyping technique that harnesses the power of open-source software. pages 1–19, 2007.

[12] M. C. Jones, I. R. Floyd, and M. B. Twidale. Patchwork prototyping with open source software. In K. S. Amant and B. Still, editors, *The Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives*, pages 126–140. Idea Group Inc (IGI), 2007.

[13] M. C. Jones and M. B. Twidale. Mashups and cscw: opportunities and issues. In *CSCW*, pages 1–4, 2006.

[14] J. Nielsen. Paper versus computer implementations as mockup scenarios for heuristic evaluation. In *Proceedings of the IFIP TC13 Third International Conference on Human-Computer Interaction*, page 320. North-Holland Publishing Co., 1990.

[15] T. O'Reilly. What is web 2.0: Design patterns and business models for the next generation of software. *MPRA Paper*, 2007.

[16] J. Palfrey and U. Gasser. Mashups interoperability and einnovation. *Berkman Publication Series, Harvard University Research Center of Information Law and University of St. Gallen, St. Gallen*, pages 1–33, 2007.

[17] ProgrammableWeb. Programmableweb - mashups, apis, and the web as platform. `http://www.programmableweb.com`, 2009. visited 06.12.2009.

[18] J. Rudd, K. Stern, and S. Isensee. Low vs. high-fidelity prototyping debate. *interactions*, 3(1):76–85, 1996.

[19] C. Shapiro and H. R. Varian. Information rules: A strategic guide to the network economy. 2000.

[20] T. Tullis. High-fidelity prototyping throughout the design process. In *Proceedings of the Human Factors Society 34th Annual Meeting (Santa Monica, CA, Human Factors Society 1990)*, page 266, 1990.

[21] M. B. Twidale and I. R. Floyd. Infrastructures from the bottom-up and the top-down: Can they meet in the middle? In *Proceedings PDC*, pages 238–241, 2008.

[22] M. van Harnielen. Exploratory user interface design using scenarios and prototypes. In *People and computers V: proceedings of the fifth conference of the British Computer Society Human-Computer Interaction Specialist Group, University of Nottingham, 5-8 September 1989*, page 191. Cambridge University Press, 1989.

[23] M. Walker, L. Takayama, and J. A. Landay. High-fidelity or low-fidelity, paper or computer choosing attributes when testing web prototypes. In *Human Factors and Ergonomics Society Annual Meeting Proceedings*, volume 46, pages 661–665. Human Factors and Ergonomics Society, 2002.

[24] N. Zang and M. B. Rosson. What's in a mashup? and why? studying the perceptions of web-active end users. In *VLHCC '08: Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 31–38, Washington, DC, USA, 2008. IEEE Computer Society.

Evaluating Prototypes for Web Applications

Korbinian Huff

Abstract— In this paper the main methods to evaluate a prototype of a web application are introduced and their characteristics are classified. These are the methods discussed: One can evaluate the prototype by analyzing the web server log data of a web application, observing user interaction with the application with the help of tracking scripts, with user tracking software, video cameras or eye tracking devices. Other methods discussed include questionnaires, interviews and the evaluation by an expert panel. All these methods are classified according to the technical complexity, the required manpower, the location where they can be deployed, the required number of test users, the financial means needed and the type of approach (practical / theoretical). Additionally the kinds of information that can be gathered with each method are introduced and examples which problems can be evaluated using which method are given. Finally an outlook on the future of web application evaluation is given.

Index Terms—Prototyping, Evaluation, Usability, Web Application

1 Introduction

Prototyping is a key factor in the development process of web applications. Prototyping means, that a preliminary model of an application is being developed which simulates parts of the final application. These parts might be incomplete in design and/or function and are solely used for evaluation. In iterative software development processes, these prototypes are then further developed using the information from the evaluation process. Subsequently these advanced prototypes are then re-evaluated until the application is feature complete and the evaluation results are satisfactory. While web applications can help users to achieve their goals, the usability of an application is a key part for the application to be accepted by users. Evaluation for usability has been shown to be useful "in terms of increased sales, increased user productivity, decreased training costs and decreased needs for user support" [2]. Therefore prototypes need to be evaluated and then refined according to the evaluation results [5].

The following evaluation methods will be discussed and classified in this paper:

- Analysis of web server logs
- Adding tracking scripts through a proxy server
- User tracking software
- Watch users using the prototype
- Observe users views with eye-tracking devices
- Evaluation by the means of questionnaires
- Evaluation by the means of face-to-face interviews
- Evaluation by an expert panel

In the following sections the different methods that can be used for the evaluation of prototypes of web applications and the different kinds of evaluation results that can be attained by the use of each method are introduced. In the following section these methods are subsequently classified and traded off against each other. Finally a conclusion and an outlook on future evaluation methods are given.

- *Korbinian Huff is studying Media Informatics at the University of Munich, Germany, E-mail: korbinian.huff@campus.lmu.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009/2010*

2 Analyze Web Server Logs

The analysis of web server logs is a very simple way to evaluate a web application. As this analysis relies on web server logs it cannot be used to evaluate prototypes of standard applications, but only prototypes of web applications.

2.1 Technical information

Standard web servers usually automatically create server logs. These logs contain information about each user query to the server. More specifically, for each query the sender's IP address, the sender's userid (if he is logged in), a time stamp, the exact request, the server status and the size of the returned data is logged [7]. Additionally the referring site and information about the user's operating system and browser are logged. A sample log entry for the request of the site "index.htm" looks like this:

```
  127.0.0.1 - - [03/Dec/2009:14:42:12
+0100] "GET /index.htm HTTP/1.1" 200 13990
"http://www.referingsite.com/links.htm"
"Mozilla/5.0 (Windows; U; Windows NT 6.0; de;
rv:1.9.0.15) Gecko/2009101601 Firefox/3.0.15
(.NET CLR 3.5.30729)"
```

2.2 What can be Evaluated?

While this log contains no information on what the user does exactly in the web application itself, it can still help to evaluate the web application, because the sequence of requests alone can tell much about what the user did. With the log data a user's visit can easily be retraced from the beginning to the end. Even some clicks of the "back" button in the browser can - indirectly - be retraced using the logged data of the referring site [16, 5].

Therefore developers can find starting points to improve their prototype, for example when a lot of users just load the first page and then leave the application that might be a sign that the first page is too confusing for the user to even bother clicking through. As server response codes are logged too, these might help in finding application errors. For example when a lot of users encounter a "404 file not found" error that might be an indication that some link in the application is not set correctly. In that case the logged referring site might help to find out from where in the web application the user came when he encountered the problem. As requests for each single element can easily be summarized it is also possible to find out, which parts of the application are the most used and thus where the developers should focus on. As the time for each request is logged, it can be retraced where in the application the users spent the most time. If this is for example a registration site the evaluation result could be to simplify the registration process. Obviously staying on a page for a longer time could also mean a user was just distracted for example because he got a phone call. Therefore a high number of server logs is needed to average the data and to get

valid results to almost any of the questions that can be answered using this method.

Also implicitly saved is the last page of the application the user has used. Knowledge of this exit page can also help for the evaluation of the web application: If the exit page is for example an order confirmation page this might mean that everything went well and the user just finished the use because he was done. If on the other hand the exit page was a page where the user had to make a decision and instead of making that decision he just exited the application, that is a sign that this step might need improvement. In order to properly evaluate which the last page the person visited was however it is required to properly identify all requests from a single user in the logs. As the userid in the logs is usually empty, because most web applications use another login mechanism than the web server's standard one, this is usually done using the IP address. In some cases though, this address is not enough to properly identify a single user because for example if multiple people from a company use a shared Internet connection they all have the same IP address. Therefore this analysis cannot be done for every single user and as it is not possible to see from the logs which log data is affected by this problem and which is not again a great number of logs is needed in order to eliminate the effects of the problem. [5]

2.3 How can this be Evaluated?

All this information can easily be evaluated using designated software that computes these findings from server log data. From free and open source software like AWStats [6] or Webalizer [3] to more complex commercial solutions like Urchin Software [9] a large number of programs is available for this purpose. The test users then just need to use the web application like they normally would [8].

3 Add Tracking Scripts Through a Proxy Server

A more complex method for the evaluation of web applications is to deliver them through a proxy server that adds tracking scripts to the web pages. While the setup and realization is more complex, also more - and different - information can be gathered with this method. Therefore other problems of the web application can be assessed using this method. Just like the previous method this method relies on web technology for gathering information and can therefore solely be used for prototypes of web and not for normal applications.

3.1 Technical Information

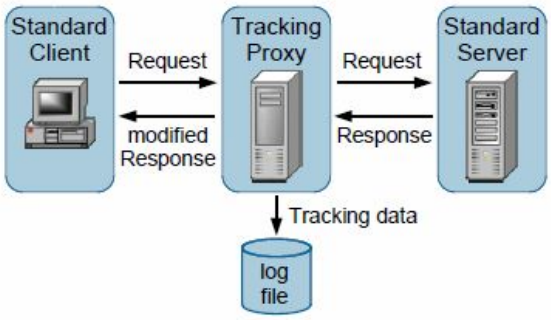


Fig. 1. Use of a proxy server [1]

This method focuses on the things the user actually does in a web application, like filling out a form, moving the mouse or clicking on something. This information is not automatically transmitted to the server, so additional measures are needed. Nowadays most web browsers are Java Script-capable, so the information can be gathered using Java Script that is added to the pages of the web application. This Java Script can save any mouse movements, mouse clicks, the size of the browser window and keys pressed on the keyboard. This information is subsequently being transmitted back to the researchers. As adding Java Script in each page of the application would induce a high effort on the server-side another approach is used: As illustrated in *figure 1*, a proxy server is used as intermediary between the client and the server. This setup ensures that only a small change has to be made on the server-side and nothing has to be changed on the user's side. This proxy server simply adds a single line of code to every html site passing through that adds the script code to the page. The gathered data is then sent back to the proxy server in regular intervals, where it is stored in a log file that can later be analyzed. Naturally a standard log file with the information discussed in the previous section is also created. The website itself is not changed in any way by the added Java Script. Even already existing scripts on the page still work after adding the tracking script. Yet, the script is limited to things happening within the web application. If a user uses a browser menu or opens a new tab this as well as anything happening outside the browser like opening another application would not be recorded. [1]

3.2 What can be Evaluated?

Using this method the data discussed in the previous section can be merged with the new information that is gathered with the Java Script. This leads to a detailed picture of what exactly the user did in the application and when. In addition to the questions that can be answered using conventional server logs, other questions can be answered with this method. Using the mouse movements it can be evaluated if the user immediately found the menu options he was looking for (straight mouse movement to the items) or needed some time to find it (slow, searching mouse movements, possibly hovering over other menu items). This can help evaluating the navigational structure of a web application. This approach uses the assumption that there is a direct relationship between the gaze position and the cursor position on the screen. Previous research shows that this relationship exists, meaning that the probability that someone is looking at the part of the screen where the mouse cursor is, is high, (more than 75%) though the number varies under different settings [4]. Using this information one can also create a visualisation of the mouse movements in a web application by overlaying the mouse tracks on a screenshot of the application. An example for such a visualization is shown in *figure 2*. Such a visualization can help to evaluate which information in an application is noticed by the user and which is not. For example *figure 2* shows, that the user did not notice the information in the top right of the application. Depending on the kind of information that is located there, this could tell the developers to move the information somewhere where it would be noticed by more users.

As the Java Script logs each mouse click and key press inside the web application, this method also logs actions that would not have been recorded by conventional server logs as they do not result in server requests. An example for this would be a click to hide some information in an application. This process can happen locally in the user's browser without any server interaction. However this information is important for the evaluation, because it might help the developers to find out which information can be hidden from the beginning in order to have a cleaner user interface without loosing information that is important for the user. [1]

3.3 How can this be Evaluated?

These results can be gathered using designated software for the proxy server like the UsaProxy that has been presented by Atterer et al. [1] or the WebQuilt system [11]. This software not only collects the data, but also processes it and offers reports that can then be used for evaluation. The test users can then use the web application just like they normally would and their actions would be recorded. Due to the high extend of possibly private data that is collected from users, it has been suggested to first ask for the users permission to record the data in order to deal with possible legal and ethical problems of the data collection [1, 5]. This can happen either within the web application itself or in a popup window [1].

4 User Tracking Software

Even more information on the usage of a web application can be gathered using user tracking software. This tracking software is installed

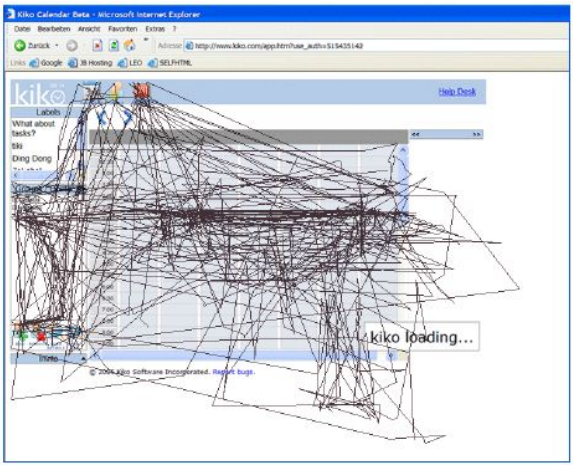


Fig. 2. Mouse trails on a screenshot of the web application [1]

on the test-users computers in order to record user interaction of any kind.

4.1 Technical Information

The method relies on installing software on computers of test users. Unlike the proxy server method, this software can then be configured to record virtually anything that happens on the computers and is not limited to user actions within the web application itself. After recording the information the applications sends it to the developers. The advantage of this method is the huge amount of information that can be gathered. While for example for the experiment of Goecks and Shavlik only the number of mouse clicks, the scrolling and mouse activity are recorded [8], for other scenarios many other things can be logged as well. Like with the proxy server method, any key presses could also be recorded and additionally only this method can for example log when another application is started. User tracking software also allows the developers to record screenshots or even video of what is happening on the screen during the test. With this method, the amount and kind of data collected can be precisely adjusted to match the developer's intentions about what needs to be evaluated [5].

4.2 What can be Evaluated?

The main difference to the data that can be collected using the proxy server method is, that using this method not only things happening within the web application can be recorded, but everything happening on the computer. For example if a user would run the calculator application to do calculations based on information in the web application this would also be recorded and could be used by developers to improve the prototype by giving the user the possibility to calculate these information within the application. Additionally with this method it can also be retraced if for example an user opens an external search website to find information in the prototype. This helps in the evaluation of the web application because it shows the user was not content with or could not find the search option in the application. [1]

4.3 How can this be Evaluated?

The method can be used remotely, but requires users to install the application before taking part in the test. Therefore the gateway hurdle for the user to take part is higher than when using a proxy or server log based method [1]. Depending on the amount and type of information recorded the analysis of the data differs. While some kinds of logged information can easily be summarized and evaluated using statistical software, other information like screenshots or videos need to be evaluated manually [17].

5 Observe Users

A prototype of a web application can also be evaluated by watching a user use it. This can give developers an idea on how users approach certain problems and help them to evaluate how useful and how easily operable the prototype is. As these methods, in contrary to the earlier discussed methods, do not require a technical implementation, all following methods can also be used with paper prototypes.

5.1 Just Observe Users

A fairly easy method to evaluate a prototype is, to watch a user interact with it. This can be done by simply observing a user and, where necessary, by taking notes. Usually however this is done with the help of at least one video camera [19]. This enables the developers to be able to exactly recreate every single step a user performed. Normally users are given a specific task to fulfill in the application [1], but depending on how advanced the prototype already is, it is also feasible to just let the user explore the application on his own. To gain additional insight on the actions, users can also be asked to explain their actions while they perform it [12].

5.2 Watch Users with Eye-Tracking Devices

Eye trackers are devices which enable application developers to retrace the direction of the view of users while using the web application. There are two major technical implementations to realize eye trackers: head-mounted and stationary cameras. The former are cameras that are mounted on a helmet, that capture the eye movement of the user. The latter option realizes this by filming the user's eyes from a stationary location. Usually this means having one or, for enhanced precision, more cameras mounted near - usually under - the screen the user uses the prototype of the web application on [13]. This setup makes it possible to retrace where a user looked during the use of the web application, but the technical equipment has the disadvantage that it creates a relatively artificial environment for the tester which might result in different behavior than in non-test situations [5].

5.3 What can be Evaluated?

Observing users can help to evaluate general usability problems with the prototype. The possible results include all that can be acquired by previous methods and more. By watching a user, observers can for example easily notice when a user is stuck somewhere in the application or when he's frustrated by a certain aspect. Obviously the limitation for this aspect is, if the user is not explicitly expressing his feelings they cannot be recorded [17]. Therefore the method of interviewing the user is often used afterwards, to gain insight on not so obvious feelings. This method is particularly suitable for the evaluation of device-specific problems with the prototype, especially because these problems would often not be recorded using other methods. For prototypes of web applications device specific feedback is however very important as web applications can be used on numerous different web-enabled devices. If for example a web application would only be poorly usable on a mobile device without touch input, previous methods would, if at all, record that the user exits the application within a short time frame from accessing it after pressing a few keys. Using this method developers can observe, that the user has problems to navigate through the application using only the arrow buttons on the device. Using this method it can for example also be evaluated, that the application is not working the way it should, because the hardware does not support Java Script or that Java Script is disabled. Using only log based methods, this would not have been noticed, because when a log entry says the user clicked the menu button the developers would automatically assume, that this opened the actual menu without knowing, that something prevented this. [14]

While these and many other things could also be evaluated using the eye tracking method, the method usually focuses on the positions where test users are looking, because the method is way to elaborate to be used for simpler tasks. Therefore the evaluation concerns mainly the layout of the application. The tracking of the views makes it possible for the developers to see which parts of an application the user noticed and which information he missed. While this evaluation can also be made using methods that log the movements of the mouse cursor, these methods all just have a certain (although high) probability of predicting the user's views [4], while this method returns definite

results. The method also shows where a user would have expected the information he was looking for. This information can help developers to evaluate the prototype by comparing this with the place where the information is actually located. It can also easily be evaluated which information on a site the user was looking for. As shown in *figure 3*, the most important information for the user in the example is located on the bottom of the page. If that result is consistent in the evaluations of multiple users, in a later prototype this information should be located near the top of the page in order to help the user to locate it. [14]

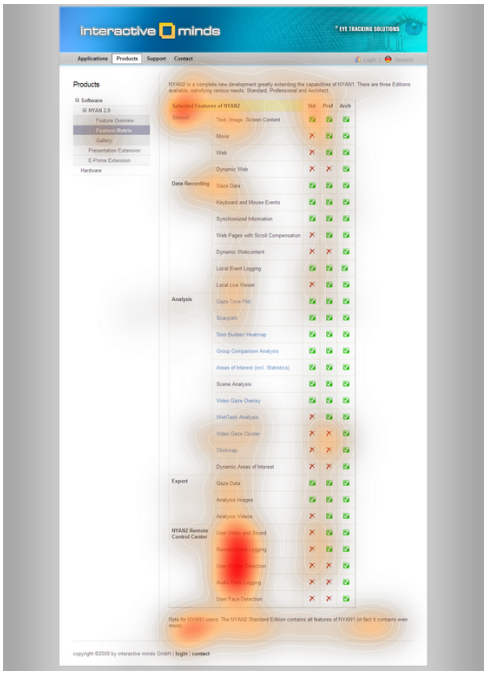


Fig. 3. Heatmap created by eye tracking software [12]

5.4 How can this be Evaluated?

Videos and records of sessions where test users use the prototype can best be evaluated by looking through them and listing and summarizing the problems the users encountered.

Traditionally the mapping of eye movements to the web application was done by hand. In the meantime software was introduced to simplify that process. First with software that provided templates that allowed to speed up the by hand mapping process then by software like the WebEyeMapper [18] that automates the whole mapping process. The evaluation itself can then be simplified by heatmaps that show graphically where a user looked. An example for such a heatmap is given in *figure 3*. These heatmaps and other analysis results and diagrams can be automatically created by designated software like NYAN [14].

6 Evaluation by Asking a User

Some information cannot be attained by simply observing a user. Therefore other methods are used to obtain information about a user's intentions, feelings and thoughts when using the web application. This subjective information can be useful to understand how a user approaches certain problems and the application itself. Users are questioned in order to give developers an insight to their perspective [17]. To obtain such information two methods can be used: After they used the prototype, the users can be asked to fill out questionnaires or the users can be asked directly in face-to-face interviews.

6.1 Evaluation by the Means of Questionnaires

Users can be asked to fill out questionnaires that ask them to report their experience with the prototype. These questions can be asked in paper based or in web based form. That and the fact that the questions asked can be precisely adjusted to match the possible problems of the web application make this method very flexible and usable in various situations. Closed questions are used to rate specific aspects of the application for example on a likert scale. From the answers to these questions absolute data can be computed to assess and compare the certain aspects. No or only poorly, absolute data can be gathered from open questions. Therefore these are mostly used for suggestions or wishes of the users. While in paper based questionnaires the questions asked to a user are predefined and can at most be skipped if they are not applicable to the user, online based questionnaires can be more flexible: Depending on previous answers of the user, different follow-up questions can be presented. Nevertheless all variations have to be included in the software from the beginning.

6.2 Evaluation by the Means of Face-to-Face Interviews

In face-to-face interviews that limitation does not exist and the interviewer can fully adapt the questions asked if the situation arises that this seems beneficial. Although, this and the fact that usually open questions are used in interviews, leads to the disadvantage that few absolute and thereby comparable data is gathered using this method. On the other hand being able to ask a user to further elaborate on something he said can be necessary for the understanding of his remarks. Also an interviewer can rephrase and explain a question if it was misunderstood by the interviewed person. Hence the method is very flexible and can be used with different questions to evaluate numerous different aspects of web applications. The data collected in interviews largely depends on the interviewer. Therefore special knowledge is needed to conduct interviews in a way that produces reliable data rather then unwillingly influence the user to adapt a certain opinion. [17]

6.3 What can be Evaluated?

Although these methods certainly have disadvantages, they are necessary and important methods, as using these methods, information can be gathered, that cannot be gathered by other methods [17]. While individual problems can be addressed with other methods, things like the overall satisfaction with a web application can be the key factor that decides if a user uses the application or rather a competing product. If for example there are only so much funds to add one feature to the application, it can be valuable to know that, while all possible features have their advocates, one feature would increase the overall satisfaction more that the others. Other topics like the feeling of an user while using the application, can help to evaluate that, for example, the user interface is perceived as intimidating or that the colour composition of the application causes certain feelings which might or might not be wanted.

Only these methods offer the possibility to get direct suggestions for improvement the developers might not have thought about. Using questions like "do you have anything else to add?" users can even express their opinion on parts of the web application that were not part of the initial evaluation. The interviewed users can also be asked to make suggestions for additional features that they would like to see in the application.

6.4 How can this be Evaluated?

Answers to open questions and generally answers given in interviews without a predetermined scale can generally best be evaluated by simply reading every single answer, as a summarization of the results is often hardly possible due to the wide array of answers. For some questions though, a summarization using generalizations like "most users mentioned user interface elements" can help. [17]

Data gathered using these generalizations and obviously already absolute data gathered in questionnaires can then be further evaluated. Therefore standard spreadsheet programs like Microsoft Excel can be used as well as designated statistical software like SPSS. There are also designated software products just for the analysis of questionnaires regarding the usability of software. One example for such a software is SUMI [15].

7 Evaluation by an Expert Panel

As other methods are often rather complex and can not be completed in a short time frame, the method to evaluate a web application by an expert panel has been introduced. This expert panel can assess the web application in a timely fashion and still point out where a prototype needs improvement.

7.1 Method Details

Using this method means first putting together a panel of experts that will then evaluate the web application. In order for the method to provide accurate results the experts need to be familiar with the matter of usability evaluation. The usability experts are then asked to assess the application in several categories according to a given checklist. Popular checklists used include the Keevil Usability Index [10] and the Web Usability Index [10], but depending on the requirements of the evaluation own criteria can be used as well. For example the Keevil Usability Index contains questions in regard to finding, understanding and presenting information as well as questions in regard to if the information is complete and helps the user to perform a task. Using the Keevil Usability index questions like "Is new information indicated?" can be answered with yes, no or with not applicable. Other, more advanced, checklists give the experts a scale to evaluate the question instead of a yes and no option in order to get a more differentiated answer. The Web Usability Index uses a numerical scale from one to five. The checklists are the main issue of the method. For example if the questions leave room for interpretation different experts might gather contradicting results.

7.2 What can be Evaluated?

The topics that can be evaluated and the problems that can be found are largely dependent on the checklists used. Examples from the Keevil Usability Index are given in the last section. The Web Usability Index [10] covers similar topics: navigation, interaction, quality and up-to-dateness, design and accessibility. These categories can be used to evaluate any prototype of a normal web application, but other categories can be included, if desired. The usual approach for this method is to evaluate the own web application and one from a competitor. Thereby a direct comparison can be made. The different categories in the Web Usability Index enable developers not only to compare the overall results but also the results in each category.

7.3 How can this be Evaluated?

Using the Keevil Usability Index [10], the result for a web application is computed by dividing the number of "yes" answers by the number of the sum of "yes" and "no" answers. The result is a percentage that represents the overall result of the evaluation for the web application, where a perfect application would reach 100%.

Using the Web Usability Index [10], a more detailed result can be computed using the following formula:

*(sum of question values - number of answered questions)/((number of questions - number of questions answered with n/a) * 4) *100*

The result is a percentage that indicates the number of usability problems. Therefore a perfect application would reach 0%. As the Web Usability Index has different categories, not only a overall result but also results for every single category can be computed using the same formula.

8 Classification of Methods to evaluate Web Applications

There are multiple ways to classify the introduced methods. The following classification can help in finding the appropriate method for different situations.

8.1 Complexity

The complexity to deploy a certain method is a key factor when choosing a method for prototyping. The deployment complexity includes the technical means and the required manpower that are required for the realization of the method. Therefore the complexity is also a main factor regarding the financial means needed.

8.1.1 Technical Complexity

For the analysis of web server logs the complexity is relatively low which makes this method rather simple to deploy. That is because the required data for the evaluation is usually already being gathered automatically and just needs to be evaluated with analysis software. As on the server-side, no changes are required on the user's side: The test user can use the application just like he would use it when he's not testing it.

Compared to that, the complexity of using tracking scripts is higher, because this method needs an additional proxy server with special tracking software. This means small changes are necessary on the server-side, while again none are necessary on the user's side. [1]

When using user tracking software, the hardware requirements on the developer's side are relatively low, as only a web server to upload the results to and the tracking software itself is needed. Nevertheless the tracking software has to be elaborately configured to record enough but not to much information, therefore the overall technical complexity is average.

Users can be observed using the application without any device other than the device they use the application on. Nevertheless to improve traceability of the results these sessions are often not only recorded on paper but also on video. Therefore at least one video camera is needed. [1, 19]

The technical complexity when using the eye tracking method is fairly high, because workstations equipped with eye tracking devices as well as special software to map the gathered data to the application are needed [18].

The technical complexity of using the evaluation method of questionnaires is low, as only paper based questionnaires or designated web-survey web applications are needed. These applications already exist in great numbers and have rather low technical requirements on the required server.

For face-to-face interviews with test users the technical complexity is very low as only a standard workspace is needed.

If the evaluation is done by an expert panel, the technical complexity is also very low, as theoretically only one workspace without any special technical hardware is sufficient.

8.1.2 Required Manpower

To analyze web server log data or the data gathered by proxy servers or the data gathered by the use of user tracking software, the required manpower is very low as the gathered data can theoretically be analyzed by just one person.

When using the methods that observe users, the required manpower is high as all test users that come to the test lab need to be introduced to the setup/technology and need to be supervised throughout the test [1]. Additionally for the eye tracking method the technical setup is complex, so specially trained people need to be employed.

For the questionnaires method, obviously more manpower is needed if paper based questionnaires are used, because the information on paper needs to be digitalized in order to analyze it. Nevertheless the required manpower for this method is rather low, as the following analysis of the data can again be done by just one person.

Face-to-face interviews on the other hand need a high amount of manpower, as obviously interviewers are needed to conduct the interviews themself.

If an expert panel is used for evaluation, the required manpower is limited to the experts and a person to analyze the data. Though, the method can only be done with properly qualified experts, and the task to find them might add to the complexity of this method. [10]

8.2 Remote or On-Site Deployment

When choosing a prototype evaluation method it is also important to know where a certain method can be deployed. While some methods can be deployed remotely, others require people to come to an on-site location. On-site setups have the advantage that more environment variables can be influenced by the researchers, for example by supplying the same hardware to all test users hardware problems can be barred and by using a bare room for the test, distractions of testers

that might result in different behavior can be reduced. Obviously these factors can also be a disadvantage for some evaluations, because test users are taken out of their normal behavior patterns and might behave different in these rather artificial environments. [5]

When using the method of analyzing web server logs, this method is considered as remotely deployed, as the test users can interact with the web application and thereby test it from wherever they have Internet access and do not have to come to a designated location [1].

The same applies to the use of the method that uses proxy servers to add tracking scripts to web pages. The data gathered by this method is also gathered remotely and the test users can access the web application from wherever they choose to. [1]

Data collected by user tracking software can also be transmitted from a remote location so that test users can test the web application from a location of their choosing.

In contrary, the eye tracking method and the method to simply observe users requires test users to come to a designated location. Therefore this method is considered to be on-site.

Questionnaires can be answered remotely whether they are paper or web based.

Face-to-face interviews are usually conducted at a designated location the testers have to come to. The method is therefore considered as on-site.

An expert panel can test the web application remotely and afterwards submit their findings to the researchers. Though it is also feasible to perform these tests on-site, as due to the small number of experts only few resources are needed on-site and remote testing would require a setup that would make the web application accessible from the outside, which might not be desired for various reasons.

Nevertheless all methods that can be deployed remotely can theoretically also be deployed on-site if for any reason that is desired. Similarly the use of eye tracking devices and the conduction of face-to-face interviews and user observations could theoretically also be done at the testers homes or other locations of their choosing, though this is usually not done due to the much higher effort needed.

8.3 Number of Users

When planing an evaluation it is also important to know how many test users are needed for which method. Methods that use small numbers of users can often be cheaper and easier to deploy, while the information gathered through methods that need high numbers of test users can be more universally valid.

For the method that uses the server logs, a high number of test users is needed, as only very few information is gathered from each user. That means that the information might not always be unambiguous, so more test data helps to point out where real problems are and where for example a user just might have been distracted and therefore needed longer for a certain step. The fact that for this method no changes have to be made on the user's side also make this method ideal for a high number of users. Also as only few data is gathered from each user even high numbers of users will not complicate the analysis of the data by much.

More data is gathered from each user using the evaluation method that adds tracking scripts through proxy servers. Therefore for this method fewer test users are required. Nevertheless as also for this method usually no changes on the user's side are needed, information from a great number of test users can also be easily gathered. As if the number of test users is too high, the analysis of the data gets increasingly complicated, a medium number of test users is usually needed.

User tracking software gathers not very different information, so a comparable number of testers is used for this method.

When using the eye tracking method all test users have to use the devices one after another [1], so a high number of testers is not feasible. Also for this method and the method of just observing people, the personnel expenditure is fairly high. Therefore comparatively few users are observed with these methods.

The results of questionnaires are usually the better the more people answered them. That means the overall number of people needed for this method is medium to high.

Naturally interviews require rather high amounts of manpower. Therefore this method is only reasonable for small groups of testers.

Using an expert panel means no test users are needed at all. This method relies just on the experts. [10]

8.4 Cost

The cost to deploy a certain method can be the main factor when choosing a method to evaluate a prototype of a web application and is closely linked to the complexity of the method used, because the technical requirements and the required manpower usually accounts for the highest costs. Additionally factors like the number of participants in the test and if the method can be deployed remotely or test users have to come to a specific location influence the costs, as more users usually means more effort and for on-site deployment a designated space for the tests is needed [1].

As both the technical and human resources requirements are low when using the method to analyze website log data, the overall cost of this method is low.

As for the proxy server method the human resources requirements are as low, but because the technical requirements are slightly higher the overall cost of the method is mean [1].

The cost for the user tracking method is made up of the relatively low costs on the technical side and the also relatively low staff costs. Nevertheless the method requires the users to install the tracking software on their computers instead of doing nothing in particular (web server logs) or simply clicking "yes" (adding tracking scripts through a proxy server), so an incentive to install the software might be needed.

Simple observations of users interacting with a prototype actually cause fairly high costs, as much supervising staff is needed and users have to come to a designated location for the evaluation.

Usage of the eye tracking method is rather complex and therefore the costs for it are high. Both the high technical complexity and the high personnel expenditure are the reasons for this. Additionally the time consumption for the test users is high and as the method cannot be deployed remotely they need to travel to the test location, so considerable payments for them might also be required.

For both, the paper based and the web based questionnaires, few people are needed and the technical requirements are also mean, so the overall cost of the method is mean.

Face-to-face interviews are more expensive, because for this method, the personnel expenditure is rather high. Just like with the eye tracking method, payments for test users might also be needed, as the time consumption for them is significant and they need to travel to the interview location.

The main part of the costs for the expert panel method is the payment for the experts. If external experts are used this payment may vary, according to the needed level of specialization of the experts. If available, experts from the own company can also be used to save money. Therefore the overall cost of this method is low [10].

8.5 Theoretical / Practical Approach

Web applications can either be assessed using a practical approach, where users directly use the application and it is recorded and analyzed how they used it and where they might have had problems. Alternatively a rather theoretical approach where experiences and requirements of users are prompted with questions can be used. Another option for a theoretical approach is to assess the web application according to best practices and evaluate where problems are, by analyzing where the application differs from these. [10]

The methods that use the practical approach are the analysis of web server logs method, the method that adds tracking scripts to the application with proxy servers, the method that uses tracking software, the method that requires observing people using the prototype and the method that uses eye tracking devices. All of these have in common, that test users use the actual web application and researchers assess the application according to the data gathered during actual usage.

Table 1. Classification overview

Method	technical complexity	required manpower	remote/ on-site	number of users	cost	approach
Analysis of web server logs	low	low	remote	high	low	practical
Tracking scripts	average	low	remote	medium	medium	practical
User tracking software	average	low	remote	medium	medium	practical
Observation of users	low	high	on-site	low	high	practical
Eye tracker	high	high	on-site	low	highest	practical
Questionnaires	low	low/medium	both	medium/high	medium	theoretical
Face-to-face interviews	low	high	on-site	low	high	theoretical
Evaluation by an expert panel	low	low	both	none	low	theoretical

In contrary a rather theoretical approach is used when the methods of questionnaires, face-to-face interviews or the evaluation by an expert panel are used. In the first two, the web application is evaluated by the answers to questions regarding the expectations of the program and regarding the actual usage. In the expert panel method the web applications are rated according to checklists that list what an optimal application should be like.

Table 1 gives an overview of all the classifications of the different methods.

9 Conclusions and Outlook

Of the presented methods none fits for all situations as no single one can point out all problems. Therefore for a thorough evaluation usually a combination of these methods is required to capture all various aspects. It my be feasible to start the evaluation process with a method that is rather simple to deploy like the expert panel method, then advance the prototype accordingly and use more complex and therefore more expensive methods not before the main aspects that can be found using the simple method are fixed, because it is likely that the more costly method would only bring up those bigger problems which are possibly masking smaller ones, too. Also simpler - and thereby cheaper - methods can be used to point out which parts of the prototype need further evaluation using more advanced - and more expensive - methods and which parts already perform well and do not need to be evaluated using the costly approach. Thereby the use of basic methods as a start can save money that would otherwise be spent evaluating perfectly well performing parts of the application. [16]

Due to the rising number of web enabled mobile devices with different device specifics, the increasing number of touch-enabled computers, as well as the rising number of devices that enable web surfing on television screens users expect to be able to use web applications on these devices, too. Therefore in the future more evaluation methods are needed that can capture device specific problems of new hardware products. [19]

References

[1] R. Atterer, M. Wnuk, and A. Schmidt. Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 203–212, New York, NY, USA, 2006. ACM.

[2] J. O. Bak, K. Nguyen, P. Risgaard, and J. Stage. Obstacles to usability evaluation in practice: a survey of software development organizations. In *NordiCHI '08: Proceedings of the 5th Nordic conference on Human-computer interaction*, pages 23–32, New York, NY, USA, 2008. ACM.

[3] B. Barrett. Home of the webalizer, December 2009. `http://www.mrunix.net/webalizer/ (Last checked: 01.12.10)`.

[4] M. C. Chen, J. R. Anderson, and M. H. Sohn. What can a mouse cursor tell us more?: correlation of eye/mouse movements on web browsing. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 281–282, New York, NY, USA, 2001. ACM.

[5] E. Cuddihy, C. Wei, J. Barrick, B. Maust, A. L. Bartell, and J. H. Spyridakis. Methods for assessing web design through the internet. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1316–1319, New York, NY, USA, 2005. ACM.

[6] L. Destailleur. Awstats - free log file analyzer for advanced statistics (gnu gpl)., December 2009. `http://awstats.sourceforge.net/ (Last checked: 01.12.10)`.

[7] A. S. Foundation. Log files - apache http server, December 2009. `http://httpd.apache.org/docs/1.3/logs.html (Last checked: 01.12.10)`.

[8] J. Goecks and J. Shavlik. Learning users' interests by unobtrusively observing their normal behavior. In *IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces*, pages 129–132, New York, NY, USA, 2000. ACM.

[9] google inc. Urchin-software von google, December 2009. `http://www.google.com/urchin/de-DE/index.html (Last checked: 01.12.10)`.

[10] I. Harms, W. Schweibenz, and S. Johannes. Usability evaluation von webangeboten mit dem web usability index. In *24. DGI-Online-Tagung 2002: Proceedings der 24.DGI-Online-Tagung 2002 - Content in Context*, pages 283–292, 2002.

[11] J. I. Hong, J. Heer, S. Waterson, and J. A. Landay. Webquilt: A proxy-based approach to remote web usability testing. *ACM Trans. Inf. Syst.*, 19(3):263–285, 2001.

[12] interactive minds. Eye tracking usability — interactive minds, December 2009. `http://www.interactive-minds.com/en/eye-tracking-usability/ (Last checked: 01.12.10)`.

[13] interactive minds. Eyetracker eyefollower — interactive minds, December 2009. `http://www.interactive-minds.com/en/eye-tracker/eyefollower/ (Last checked: 01.12.10)`.

[14] interactive minds. Nyan 2.0xt editions — interactive minds, December 2009. `http://www.interactive-minds.com/en/eye-tracking-software/nyan-20xt-editions/ (Last checked: 01.12.10)`.

[15] J. Kirakowski. The use of questionnaire methods for usability assessment, 1994. `http://www.ucc.ie/hfrg/questionnaires/sumi/sumipapp.html (Last checked: 01.12.10)`.

[16] N. Nakamichi, K. Shima, M. Sakai, and K.-i. Matsumoto. Detecting low usability web pages using quantitative data of users' behavior. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 569–576, New York, NY, USA, 2006. ACM.

[17] M. Q. Patton. *Qualitative research and evaluation methods*. SAGE, 2002.

[18] R. W. Reeder, P. Pirolli, and S. K. Card. Webeyemapper and weblogger: tools for analyzing eye tracking data collected in web-use studies. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 19–20, New York, NY, USA, 2001. ACM.

[19] S. Waterson, J. A. Landay, and T. Matthews. In the lab and out in the wild: remote web usability testing for mobile devices. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems*, pages 796–797, New York, NY, USA, 2002. ACM.

Evaluating Prototypes for Web Applications

Thomas Creutzenberg

Abstract— The subject of this paper is to present an overview over some possibilities to evaluate prototypes for web applications. The focus is set on different tools and methods. They are divided into separate categories. The category evaluation with checklists features the Keevil usability index and the Web usability index. Behavior-based evaluation with logging presents Web Usage Mining with WUM, WebQuilt, UsaProxy and GINIS, automated tools supporting evaluation takes a look at Quantitative Evaluation of Web Sites and Automated Evaluation with Guideline Review and the category interviews introduces the Repertory Grid Technique for Web Site Evaluation. It is described why a tool to evaluate the flow experience is missing and a new possible form of the evaluation of web applications is suggested. Then an overview over the different tools and methods is given. Finally it is concluded that checklists are the most flexible type of evaluation for web applications, automated tools supporting evaluation are best for evaluating formal aspects prototypes of of web applications and some of the tools of behavior-based evaluation with logging are best for remote evaluation in detail.

Index Terms—Evaluation, Prototype, Web Application, Overview, Keevil Usability Index, Web Usability Index, Web Usage Mining, WUM, WebQuilt, UsaProxy, GINIS, Automated Web Evaluation, Quantitative Evaluation of Web Sites, Guideline Review, Repertory Grid Technique, Flow Experience

1 Introduction

This paper aims at giving an overview over some methods and tools to evaluate prototypes of web applications as well as classifying them, although this paper does not claim to be exhaustive. The focus lies on the different tools and techniques and not so much on the underlying theories behind the approaches. Also, an outlook on possibly missing tools is given.

The structure of this paper is as follows: At first the question is clarified why prototyping is important for software development in general. Then some of the different approaches to evaluate web applications and their prototypes are presented. The tools and methods discussed are divided into different categories. Types of evaluation can be distinguished between formative evaluation, this takes place during the design process, and summative evaluation, this takes place after the product or the stage of a prototype is final [10]. Evaluation types are not useful as categories though as tools can be useful for formative and summative evaluation at the same time. But tools have different approaches to the process of evaluation. They employ different means to achieve the evaluation of web applications although they share the same theory and goal. To measure and evaluate the usability of a web application there is for example the approach to convey a survey or to log user behavior. The categories covered in this paper are evaluation with checklists, behavior-based evaluation by logging their actions, automated tools supporting evaluation and conducting interviews with the Grid Repertory Technique. After presenting tools and methods in each of those categories ordered by date of publication there is an outlook on the flow experience while interacting with a web application. A tool to measure and evaluate the flow experience by users is not developed yet. Next is a section to suggest a new form of evaluation with modular prototypes of web applications to improve their usability and layout. The next section then discusses the different evaluation tools and techniques and tries to compare them against each other. Finally it is concluded which tool is best used for which application are made here.

2 Motivation

The motivation behind this paper is to present different types of evaluation tools and methods for prototypes of web application compared to each other. Already 25 years ago Floyd [5] stated: "(...) prototyping serves to introduce into software development methodology an element of communication and feedback". Prototyping is not only done for theorem proof but also because changes and improvements can be made and tested a lot easier with a prototype before the final version of the product is presented to a large audience. It is important to evaluate prototypes of web applications even though they can be updated quite easily by just putting a new version on the hosting server because the application should be as final as possible when it is published. This is because poor web applications are turning users away according to Matera et al. [10].

Atterer et al. stated in 2006 that "in recent years, the web has constantly been gaining importance as a platform for applications" [1]. A web application is an application to be accessed via a web page with a browser. Due to this connection the evaluation methods and tools for web pages and web sites are also valid for web applications. It is redundant to say that this is also true for their prototypes.

As the web hosts a growing number of web applications in recent years and as prototyping is an integral part of software development, knowing how to evaluate prototypes of web applications can only be of advantage. The following sections will give a rough overview over some of the possibilities to evaluate how to achieve this goal.

3 Evaluation with Checklists

Checklists are a popular tool to conduct empirical studies or surveys. A checklist consists of a number of questions with one response option each. Response options can be check boxes in various quantities and meanings or the possibility to write down text or numbers. Checklists can be filled out by experts conducting an interview for qualitative reasons. Alternatively, it can be a survey in written form or an online survey which is normally used to get feedback from larger audiences to get quantitative feedback. The answers to the questions are evaluated as intended by the designer of the checklist. This knowledge can then be used to improve the web application.

There are several tools to evaluate web applications with checklists like the Keevil usability index [8] of 1998 or the Web usability index [6] of 2002 which are both described in this chapter. The Web usability index is inspired by the previous Keevil usability index [6] and tries to overcome some of its shortcomings.

3.1 Keevil Usability Index (1998)

The Keevil usability index [8] is a checklist to measure the usability index of a web site and can be used either formative or summative.

It can be obtained for free [8] in HTML or Excel format. For this tool, usability refers to finding, understanding and using the informa-

- *Thomas Creutzenberg is studying Media Informatics at the University of Munich, Germany, E-mail: thomas.creutzenberg@campus.lmu.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009*

tion of a web site. The checklist contains over 200 questions. Each question can have one answer out of Yes, No or Not Applicable (N/A). The questions are based on related research on the web and usability in general. A scoring system was intentionally not used, because a "scoring system is open to interpretation by the evaluators" [8]. Web site creators are encouraged to add extra questions about their specific web site design and purpose. The questions are organized in five different usability categories [8]:

- Finding the information: Can you find the information you want?
- Understanding the information: After you found the information, can you read and understand it?
- Supporting User Tasks: Does the information help you to perform the task?
- Evaluating the Technical Accuracy: Is the technical information complete?
- Presenting the Information: Does the information look like a quality product?

Each category contains a set of matching questions. Keevil says that using the usability index is simple, but to use it to full effect one needs to understand many usability principles *(see figure 1)*.

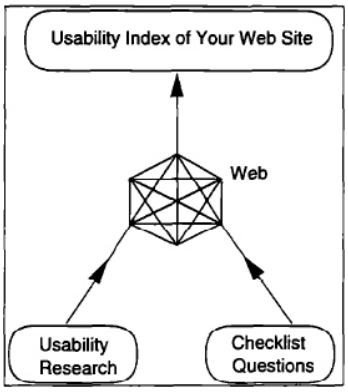


Fig. 1. Keevil usability index in context [8]

The process to measure the usability index of a web site demands four steps:

1. Displaying and Downloading the Checklist
2. Determining the Purpose and Style
3. Asking the Questions
4. Answering the questions

Either experts or normal web users can fill out the checklist. Keevil states that it is not necessary to have the web site peer-reviewed, also normal users should be able to cope with it. After the whole checklist is answered the usability index is calculated by the following formula:

$$Usability\ Index\ =\ \frac{Total\ Yes\ Answers}{Total\ Yes\ +\ No\ Answers}\ x\ 100$$

The result is the usability index of the web site in percent. Not applicable answers are not taken into account for the calculation.

In a nutshell one can say that the Keevil usability index is a simple and structured method to evaluate web sites. Additionally, it is simple and inexpensive to use. It provides a rough impression about the usability of a web site.

3.2 Web Usability Index (2002)

The web usability index as introduced by Harms et al. [6] in 2002 is a method to evaluate information-oriented web sites. It is based on the principles of the Keevil usability index [8], suggestions of related checklists and the fundamentals of web usability. It can also be used either formative or summative.

Since the introduction of the Keevil usability index the increasing use of the web as a means for communication and source of information changed the expectations and the needs of the users. The web usability index tries to consider this by adjusting it to the state of the web-related research of 2002. It can be downloaded for free from the web site of the University of Saarland, Germany [6]. It consists of a checklist with nearly 150 questions in Excel format. The web site creators are invited to add additional questions and even categories for their specific web application if necessary. By default the questions are grouped in five categories:

- Navigation and Orientation: consistency of navigation, color of links, etc
- Interaction and Information-Exchange: availability of a homepage, skip functionality for intros, etc
- Being up-to-date and Quality: marking of texts with author and date, absence of spelling mistakes, etc
- Information- and Text-Design: size of the font, expressiveness if icons, etc
- Ease of Access and Accessibility: connection between URL and web site, availability of high- and low-tech variants of the web site, etc

The web usability index is aimed to be peer-reviewed. Experts test the web application individually and fill in the Excel sheet. What is also important about it is that a second web site has to be evaluated parallel to the web site to be evaluated. The checklist introduces a scoring system for each question with a five point scale ranging from one (very good), two (good), three (satisfactory) over four (fair) to five (poor). In addition there is the option to check not applicable (N/A) if the question does not relate to the web application. After answering all questions the two web sites are then automatically compared to one another. The formula is:

$$Web\ Usability\ Index\ =$$

$$\frac{Total\ Sum\ Of\ Scores\ -\ Total\ Answered\ Questions}{(Total\ Questions\ -\ Total\ N/A\ Questions)\ x\ 4}\ x\ 100$$

The result is expressed in percent, a lower number is equal to few usability problems, a high number means the web application has a lot of usability problems. This formula also takes into account how many questions were answered with not applicable (N/A). The less questions are answered with not applicable, the greater the significance of the result. The finding shows a quick overview of the usability problems of both competing web sites. The significance of the overall score is not as important as the direct comparison between the itemized categories where the real shortcomings can be detected. But the web usability index can give no advice on what precisely has to be improved. If serious problems with usability are found additional measures have to be taken.

Summing up, by default the web usability index is tool to get an overview of the shortcomings of a web application regarding usability compared to another web site. It does not tell if it has a good usability but denotes just a comparison.

4 Behavior-based Evaluation with Logging

This section describes some of the tools to evaluate web applications according to logged user behavior. There are many different kinds of behavior-based evaluation techniques, like eye-tracking or video-recording a tester. The focus in this section though is on the evaluation

of user actions, for example clicking, scrolling, bookmarking, printing, which were logged either client-side, server-side or through a proxy server in-between the client and the server. The recorded log files are prepared and aggregated for evaluation. This data is then analyzed and evaluated to improve the web application.

Client-side logging can be done with the GINIS Framework [20], server-side logging with WUM [3, 16] and the proxy-approach is taken by WebQuilt [7] and UsaProxy [1].

4.1 Web Usage Mining with WUM (2000)

M. Spiliopoulou explains in her article "Web Usage Mining for Web Site Evaluation: Making a site better fit its users" [17] how developers can use web mining tools to summative evaluate web applications in regards to user behavior.

Data mining is a procedure to extract patterns from data, in this case web server log files, to generate information. The knowledge acquired can be used in multiple ways, one of them is to discover navigation patterns in a web site. Web logs have to be prepared for evaluation and can then be analyzed with a mining tool *(see figure 2)*. The preparation

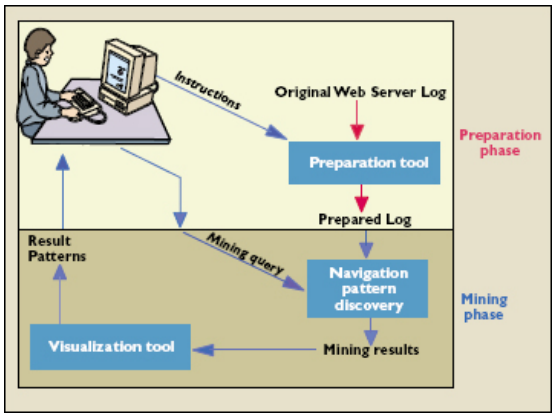


Fig. 2. Web usage mining: The process of discovering navigation patterns in a site [17]

of the web log for analysis demands to allocate the different log entries to single users. Only then does the mining process make sense. It has to be taken into account that there are several problems distinguishing between single users. For example proxys, client-side caching or security considerations taken by users with privacy concerns will make it difficult to impossible to differentiate between users due to the nature of normal web logs. Additionally, automated web spiders browsing the web site will leave traces in the log file which are not appropriate for evaluation. After the data preparation phase users' activities will be arranged into sequences of page visits. Then the mining phase can begin. The data gathered is mined for sequences and patterns. But which? To also find the interesting non-trivial sequences like entry page then main page, web log miners are needed which also understand abstract pattern descriptions. Spiliopoulou evaluated two different web log miners and but recommends the Web Utilization Miner WUM [3, 16] for several reasons. WUM is a web mining tool with special attention to increased interaction possibilities with human users *(compare to figure 2)*. It employs a powerful mining language. Experts can use their knowledge to guide the miner and thus refine or refocus the information discovery process according to previous mining results. WUM's mining language is similar to SQL queries and can understand complex requests. The result of a WUM query is a tree composed of the routes users took through a web site regarding the conditions stated in the query. With this information the developer can for example compare routes taken by the users frequently with what he intended to be frequent routes taken and adjust the different web pages accordingly. This, of course, implies that the expert evaluating the web site has a thorough understanding of usability, navigation and the web in general.

By discovering bad and popular routes through web sites among users, developers can progressively improve their prototypes of web application. To use WUM they have to have access to the web logs though and they have to learn the WUM mining language to be able to write their own queries and to modify existing ones.

4.2 WebQuilt (2001)

A tool to realize web logging with an integrated visualization system was introduced in the form of WebQuilt by Hong et al. [7] in 2001. It uses the proxy approach to overcome many logging problems with previous methods. The main goal of WebQuilt is to help developers to understand user behavior to improve the usability of their web applications. It can only be used for summative evaluation as unfinished web applications would interfere with normal user behavior thus leaving wrong traces in the log files.

WebQuilt exerts automated systems to interpret the special log files it creates. But it is designed to give a detailed overview about user's courses of navigation on a web site, thus it is a tool to evaluate user behavior. WebQuilt gathers information about the paths specific users take while they interact with a web application. Previous systems which did the same were either client-side or server-side tools which had to be installed on the appropriate machines. There are several problems with each of both approaches, ranging from compatibility of operating systems to ownership and restricted access to the machines. WebQuilt addresses these problems by employing a different approach to the data collection process. A proxy server is set in-between the server and the client *(see figure 3)*. WebQuilt can be used on any web

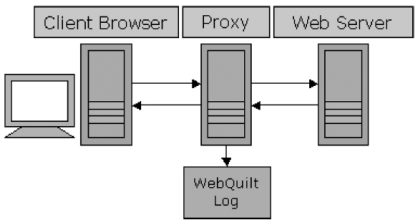


Fig. 3. WebQuilt's proxy approach [7]

site. It changes the HTML code of the web page to be displayed in such a way that any further request from this page can be exactly connected to this special user. As the proxy is in-between the client and the server and only changes HTML code there are no problems with different browsers or operating systems of neither client nor server. This takes care of the first two needs. The third need is addressed by an integrated visualization tool. It can take aggregated data from one or several test sessions and display the path users took alongside displaying the web pages themselves. Additionally to paths, also time-based metrics can be shown, for example how long a user spent in average on a certain web page. These indicators, among other things, help to identify key entry and exit points, where navigation does not work like intended and where there might be dead ends. All in all, the data gathered by WebQuilt supports designers to discover where users encounter obstacles while navigating the web application. The intended way to use WebQuilt is the following:

1. Set up a set of tasks
2. Recruit 20 to 100 participants
3. Give the participants a starting URL that uses the WebQuilt proxy
4. Ask the participants to complete the tasks
5. Use WebQuilt to collect the data
6. Aggregate, visualize and interact with the data collected
7. Spot and identify usability problems

8. Fix these problems
9. Repeat the process iteratively

It can also be used in conjunction with other evaluation tools, like for example online surveys. WebQuilt's architecture is divided into five different independent components: the proxy logger, the action inferencer, the graph merger, the graph layout and the visualization. The proxy logger captures the actions of the user in a special WebQuilt log file format. The action inferencer sets a log file of page requests into context and creates a log of the actions performed. An action can either be requesting a page, clicking the back button or clicking the forward button of the browser. Developers can write their own plug-ins for the action inferencer to address their needs. The graph merger can merge several log files if desired. This allows the aggregated evaluation of the data of several users at once. The graph layout then prepares the data for visualization. An algorithm spreads out the relevant web pages according to their connection path and importance. Visualization then displays the aggregated and converted data. Web pages are displayed as rendered screen captures of that page in a web browser. The connections and path of action is visualized by arrows in-between these screen captures. There are also many filters available to help to visualize certain relations. For example can the color of an arrow be set to represent the frequency of use of that connection. It is possible to zoom in and out of the view to get an overview or to examine single web pages in detail.

Hong et al. ([7], p. 283) summarize WebQuilt best with their own words:

> We have described WebQuilt, an extensible framework for helping web designers capture, analyze, and visualize web usage where the task is known. WebQuilts proxy-based approach to logging overcomes many of the problems encountered with server-side and client-side logging, in that it is fast and easy to deploy, can be used on any site, can be used with other usability tools such as online surveys, and is compatible with a wide range of operating systems and web browsers.

But WebQuilt also has a few limitations. It cannot handle web applications which employ JavaScript. Also, a lot of actions connected to user behavior are not captured, for example mouse clicks and key strokes, which might be important to recreate and understand the whole user session.

4.3 UsaProxy (2006)

Atterer et al. [1] introduced UsaProxy in 2006. It is a tool to extensively track user interaction on a web site. It also uses the proxy approach like WebQuilt [7] but implicates the means of modern web sites like JavaScript. UsaProxy works by adding JavaScript code to requested HTML pages before they are passed on to the client. The additional JavaScript code tracks user input and behavior and passes it on to the proxy server. The logged data can then be used for summative evaluation purposes.

UsaProxy was developed to utilize a tracking technology which is flexible yet non-invasive. The general requirements for it were to track user actions in detail, be platform independent, be transparent in operation not to alter the browsing experience, to employ as few server-side or client-side changes as possible and to work automated. To address these needs UsaProxy follows the proxy approach. All requests from the client are not sent to the web site's server but are addressed to the proxy server. Here the proxy requests the web page as asked by the client and waits for the response. The response is then modified by adding JavaScript code to the HTML of the web site and is passed on to the client *(see figure 4)*. If images are requested, no code will be added. UsaProxy not only logs all traffic to and from the client, including page visits. The client-side JavaScript code also records user actions like the movement of the mouse cursor, mouse clicks and hovering, key presses, scrolling, changes of window size, which elements are focused alongside all time based metrics. This detailed information is then passed on to the proxy server for logging. The collected

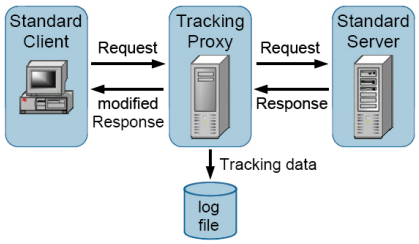


Fig. 4. UsaProxy functionality [1]

knowledge about the user's behavior can then be used in a variety of different scenarios to help to evaluate a web application [1]:

- User profiling
- Development or debugging of web applications
- Usage analysis of websites, e.g. determining detailed usage patterns for purposes of marketing, business process streamlining or similar
- Usability tests of websites
- Self-adapting websites, i.e. websites which adapt some of their content (menu structure, main text on front page) to the varying demands of users

Atterer et al. [1] state that the information gathered could be used for other applications as well. Furthermore implicit information about the user can be extracted from the log files. This obviously raises questions about privacy, ethical and legal issues. They strongly recommend to ask a user for his approval to be allowed to track his actions. For evaluation purposes, the aggregated data can be visualized in a variety of ways from normal website statistics to the visualization of mouse movements on a web page *(see figure 5)*.

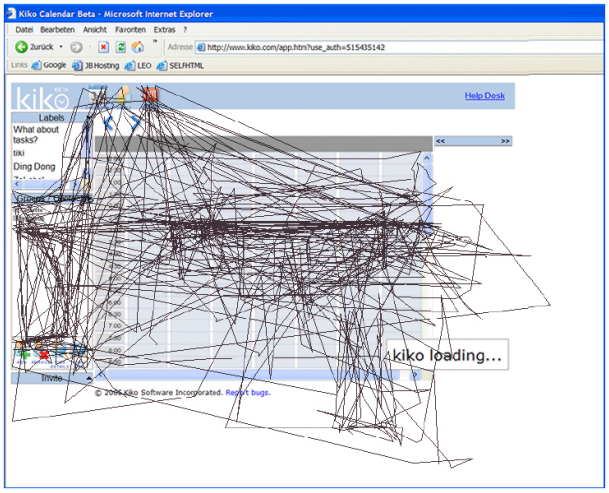


Fig. 5. Mouse trails recorded by the HTTP proxy, combined with a screenshot of the website. [1]

UsaProxy is a flexible tool to track user actions in great detail while staying transparent. Not even client-side or server-side changes to any setups have to be done. The information and the knowledge about the user behavior can then be used to evaluate a web site. But the amount of information gathered about users raises privacy concerns, UsaProxy should not be abused.

4.4 GINIS (2006)

The GINIS Framework is a client-side tool to log and analyze user behavior. It is based on customized web browser and was developed by Velayathan and Yamada [20] in 2006. The data collected is used to build a user profile according to the web pages a user shows interest in to help with the summative evaluation of web pages.

By detecting and learning user actions like for example clicking, scrolling, bookmarking or printing, it is possible to extract user profiles from this data. A behavioral interaction database can be built. Web pages can then automatically be evaluated according to these user profiles. Velayathan and Yamada state that implicit user behavior plays a major role to accomplish this task. The GINIS Framework is based on the .NET Framework 2.0. It uses many MS Internet Explorer components. Collecting user behavior is done in two separate parts. In the first stage, the learning stage *(see figure 6)*, every time a user moves

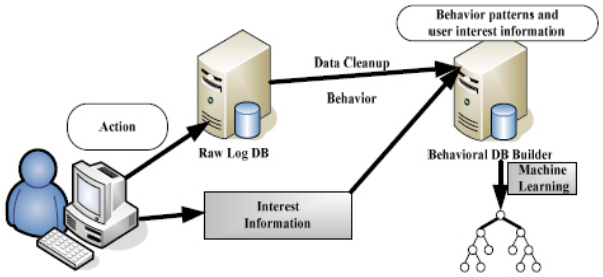


Fig. 6. GINIS Learning Stage [20]

on to a new web page, he is asked whether he likes the page or not. The default setting is "unknown", so the user can go on quickly if desired without collecting data from forced decisions. These decisions are stored in a database. An automated learning engine cleans up the raw data and stores it in the User Behavioral Database. The second stage, the testing stage *(see figure 7)*, compares the user action with

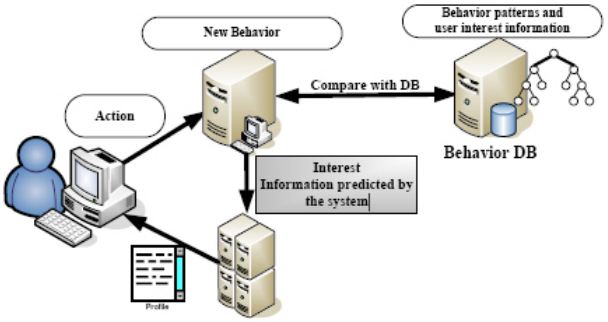


Fig. 7. GINIS Testing Stage [20]

previous recorded patterns in the User Behavioral Database. Based on the comparison of this information the user's implicit interest in a web page is predicted. The four main modules of GINIS are [20]:

1. The browser: Very similar to IE6.0 but with tabs. Logs user navigation actions
2. The logger: Logs more than 70 actions and 40 behaviors
3. The analyzer: Employs a decision tree algorithm to build a behavioral database
4. The predictor: Automatically classifies web pages into the classes "interested" and "not interested"

According to the different user profiles collected, GINIS evaluates different web pages for different user groups and applications. A big database with profiles is needed to evaluate prototypes of web applications, but a general predictions can be made.

5 Automated Tools supporting Evaluation

Another methodology to evaluate prototypes of web applications is the approach to do the evaluation automatically according to certain criteria. There are some restrictions to the automatic evaluation with software. Humans can evaluate implicit information and the context of the content. Software can only interpret the facts given like tags, number of links, etcetera, but software is much more accurate in finding and evaluating these elements than humans can be. Thus automatic evaluation is useful regarding these aspects, faster and not prone to human interpretation of the criteria given. Prototypes of web applications can only gain value by evaluating their shortcomings in certain basic usability features.

The following subsections present two different approaches to this topic. The quantitative evaluation of web site content and structure presented by Bauer and Scharl [2] describes how huge quantities of web sites can be compared, Vanderdonckt and Beirekdar [19] present an approach to evaluate a web application according to a set of freely definable guidelines.

5.1 Quantitative Evaluation of Web Sites (2000)

In 2000 Bauer and Scharl [2] introduced an approach to automatically classify and summative evaluate web sites which are publicly accessible according to their content and structure. Along other useful applications it is also possible to evaluate prototypes of other web applications with the data gathered. The system works by employing three different autonomous software tools. The first mirrors existing and publicly accessible web sites, the second extracts the classification criteria and the third uses analyzing and clustering mechanisms to aggregate the information.

Software which evaluates web sites is not prone to subjective judgment of individual human experts. Though, evaluation has to be according to a set of rules which has to be defined. But evaluation itself is a lot faster and more efficient and thus can be applied to a larger number of web sites than humans could do. Of course, the data collected is more technical and implicit information of the web applications is lost. Possible applications of the data collected can be divided into three areas [2]:

- Snapshot analysis (static): Analysis of a large number of web sites at a given time
- Longitudinal analysis (dynamic): A defined set of web sites can be documented and analyzed over a longer period of time
- Comparative analysis: Web sites are compared to evaluate and optimize the different applications

The last approach, the comparative analysis, is the one which is useful for evaluating prototypes of web applications. To actually build a database with web sites available on the world wide web, Bauer and Scharl [2] suggest these three steps:

1. Web mirroring
2. Extracting the classification criteria
3. Analysis and clustering mechanisms

Each phase has an input and output interface which is clearly defined and connects it to the next phase. This way each of the three phases can be run independently from the other phases and the tools employed can be replaced by other tools with the same interfaces if desired. Bauer and Scharl [2] recommend the free software tool HTTrack [14]. It is set to only download HTML files. Images, sound, video, etcetera are not downloaded. Bauer and Scharl [2] state that the sheer amount of memory space would not be feasible. A maximum size for the HTML files can be set as well. These statements are from the year 2000. Nowadays it should be possible to download all data for specific tests as well as today AJAX website cannot be downloaded and thus can not be evaluated correctly. The WebAnalyzer tool is written by Bauer and Scharl [2] themselves in Perl 5. It parses the HTML files saved by the

web mirroring tool. It can be set to extract classification criteria categorized into three groups: content (number of documents, kilobytes downloaded, number of file types and number of images), interactivity (number of forms, number of documents with JavaScript, number of Java applets and number of mailto-links) and navigation (frames, number of internal links, number of external links, number of anchors and number of links to anchors). In addition to this a text file is extracted from the HTML and stored with the data for textual analysis. To actually generate semantic meaning out of this raw information, the data needs to undergo the process of the analysis and clustering mechanisms integrated in the program. With a data base of many evaluated web sites in different categories, it is possible to evaluate those against prototypes of web applications of these categories [2]. The data gathered shows differences and similarities and thus can help to improve the prototypes.

The quantitative evaluation of web site content and structure is an automated process to collect information which is useful for the evaluation and classification of prototypes of web applications. The downside is that a huge data base is needed. The web is growing all the time, so there will always be more memory needed. The quality of the information gathered is only of technical nature, no statement relating to the relevance of the web application or any implicit information can be made.

5.2 Automated Evaluation with Guideline Review (2004)

Vanderdonckt and Beirekdar [19] present an approach to automatically evaluate a web site according to flexible usability and accessibility guidelines. The guidelines can be defined using a Guideline Definition Language which is implemented using a XML scheme. The simultaneous summative evaluation of several guidelines at once is also supported.

With the presented approach it is possible to automatically check web sites for freely defined guidelines. Usability guidelines are often misinterpreted by developers because of inappropriate phrasing which makes it favorable to have them checked by software. Of course, only conditions can be checked, which can be traced in the HTML code of a web application, like tags or attributes. The evaluation process itself employs the interaction of two separate elements: the evaluation logic and the evaluation engine. The separation of those two was a new approach in 2004 according to Vanderdonckt and Beirekdar [19]. This allows the evaluation logic to be autonomous from the evaluation process, both can be defined in different phases *(see figure 8)*. Structuring the usability and accessibility guidelines is the first phase of the process. To be able to formulate reasonable guideline postulates a thorough knowledge of usability and accessibility principles as well as an understanding of HTML is necessary. These two elements need to be brought together to formulate reasonable guidelines which are feasible to be applied on the data given. Formal guidelines are established and expressed in the Guideline Definition Language. This is a XML-compliant language where it is possible to formalize these guidelines for automatic evaluation. The second phase consists of three steps. The first step is site crawling. A web site specified by its URL is downloaded automatically. Any further processing is based on this data. In the second step the web page is parsed in a single scan according to the parameters given in guidelines. It can be prompted which guidelines have to be taken into account. The web page will only be parsed looking for the elements in question. The third step, the evaluation, generates a detailed report regarding the page elements which have respected or violated the guideline defined. The fundamental concepts of Vanderdonckt and Beirekdar's approach and their interactions [19] are visualized in figure 9. The Guideline Definition Language allows the evaluation of the basic data types [19]: Integer, Float, Color and String. Vanderdonckt and Beirekdar added the constructed data types: Sequence, Table and Cartesian Product, to be able to evaluate more complex conditions and to be more flexible in general. With this set of evaluation criteria it is possible to evaluate web applications regarding guidelines like [19]: "Use a limited number of fonts in a web page", "Select colors that will make your page easy to read by people with color blindness", "Web pages shall be designed

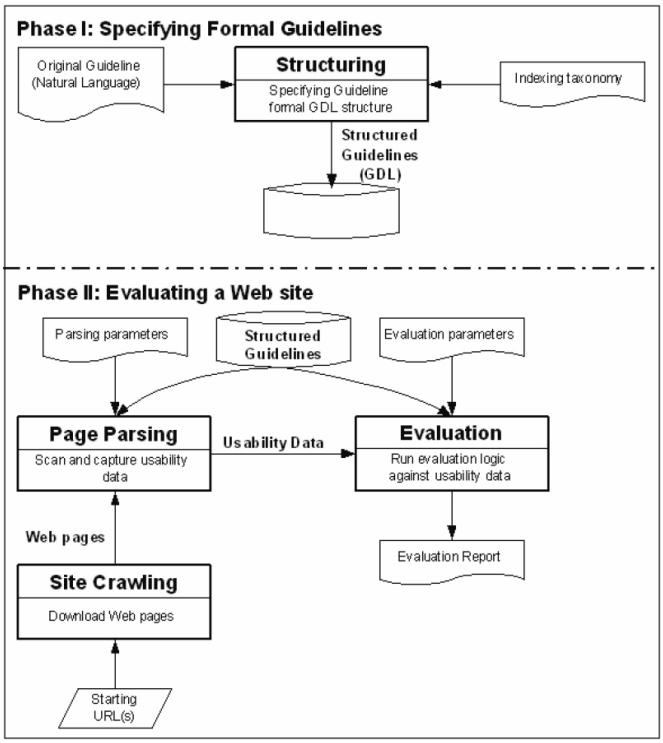


Fig. 8. The two main phases of evaluation process [19]

so that all information conveyed with color is also available using any combination of the above markup elements" (Bold, Italic, text size, text font) or to check if all elements are tagged so blind people can browse the web page correctly.

The approach presented by Vanderdonckt and Beirekdar for the automated evaluation of web applications according to guidelines is a flexible one which can also be used for prototypes of web applications. Using the Guideline Definition Language, usability and accessibility guidelines can be evaluated, as well as simple or complex self-defined guidelines. The freedom of formulating the guidelines for an automated check also makes it feasible to evaluate web applications according to established guidelines proposed by W3C, ISO or Section508 [19]. This is a good way to help to run automated tests to improve prototypes of web applications. But still, developers have to have a thorough understanding of usability and accessibility principles

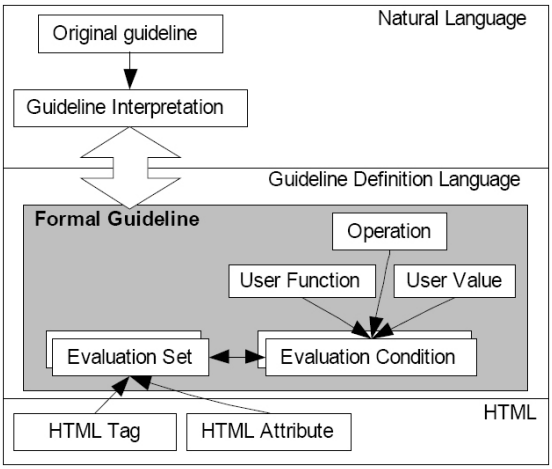


Fig. 9. Fundamental Concepts [19]

as well as a thorough understanding of HTML and XML to add own guidelines to the repertoire.

6 Interviews - RGT for Web Site Evaluation (2009)

Interviews demand that there is a person conducting the interview, the interviewer, and a person to be interviewed, the interviewee. The interviewer asks questions which are answered by the interviewee. In contrast to evaluation with checklists, interviews do have a set of questions to be asked, but also take the additional information given by the interviewee into account. With checklists, any extra information is lost for the evaluation of the given questions.

The Repertory Grid Technique (RGT) is based on Kelly's Personal Construct Theory [9] according to his Repertory Grid interviewing technique. It was first used in 2009 by Tan and Tung [18] in 2009 to explore web site evaluation criteria for commercial B2C web sites. They explain why it is suitable to use the Repertory Grid Technique for research in the area of web site evaluation [18]. The Repertory Grid Technique can be used for formative or summative evaluation purposes.

The working order for the Repertory Grid Technique in general is two steps. First, a list of concepts is generated. Concepts for web applications are for example "implementation of the search functionality", "positioning of the menu", "design to allow frequent updates", "information based on user interest". Tan and Tung [18] propose a wide variety of concepts grouped in meta-categories. The meta-categories are divided into conceptualizations which are then divided into construct classes which house the real constructs. For example the meta-category "navigation" is divided into the conceptualizations "position of menu / navigation bar", "search function", "rollover effect", "links" and "user friendliness". Each of those is further divided into construct classes. "Search function" is for example divided into "scope of search" and "quality of search". Each of those construct classes can then result into concrete constructs. Examples of constructs for "scope of search" would be "offer global search engine" and "search engine within own website". The second step is to rate these constructs on an uneven scale. Mostly used are intervals of five or seven with the Repertory Grid Technique.

The evaluation of a web application according to the Repertory Grid Technique according to the process employed by Tan and Tung would work like this [18]:

1. Sampling: Invite a number of representative subjects or experts to be interviewed
2. The Repertory Grid Interview Process
   (a) Elements Selection: Select relevant elements. In the case of the evaluation of web applications, comparable or competitive web applications are considered.
   (b) Construct elicitation: This works by "Triading" [9]. Select the web application to be evaluated and two additional ones. Now have the interviewee identify how two of these are similar or different from the third considering the appropriate design principles for web applications. This way the constructs will be generated. For each construct, ask the interviewee if a high or low value for this construct is desirable. Repeat until the interviewee does not find any new constructs.
   (c) Rating of elements along constructs: As there are no new constructs to be found, the interviewee is asked to rate all constructs gathered for web applications.
3. Analysis of Data: The information collected from the interview now has to be analyzed. If other web applications have better ratings of constructs than the one to be evaluated, it has to checked why and how it can be improved. For this task a thorough understanding of design principles is needed though.

Interviews withe the Repertory Grid Technique can identify shortcomings of web applications or their prototypes in all of their aspects. Experts are consulted and can openly create constructs of all aspects of the application which are then evaluated and compared to other web applications. This may result in aspects the developers did not think of in the first place as there are no set questions which need to be evaluated. But, inviting interviewees is time and cost consuming.

7 A Future Idea: Flow Experience

At this moment there is no tool to evaluate the flow experience while using a web application.

The "flow construct" was first introduced by Mihaly Csikszentmihalyi in 1975 [4]. He describes it as ([4], p. 6) "the holistic sensation that people feel when they act with total involvement". In 1996 Hoffman and Novak [11] define flow in online environments as:

> The state occurring during network navigation which is: 1) characterized by a seamless sequence of responses facilitated by machine interactivity, 2) intrinsically enjoyable, 3) accompanied by a loss of self-consciousness, and 4) self-reinforcing.

A lot of research and empirical studies were performed around the flow construct in connection with the web, starting with Novak and Hoffman in 1996 [11] and in 1997 [12]. Several others were conducted until the present day, to mention two: Pace in 2003 [13] or Skadberg and Kimmel in 2004 [15]. Skadberg and Kimmel try to give an answer to the question what the flow experience is for web users in information-seeking activities. Their empirical study [15] shows that experiencing flow while interacting with a web application increases the user's learning about its content. They also proof that increased learning in this context also contributes to a positive change in attitude and behavior towards the content presented. These elements are a favorable goal to achieve for any web application. To integrate the evaluation of flow to the iterative process of prototyping would be a great benefit.

Some of these studies use checklists for their empirical studies ([12], [15]), but they do not propose a universal checklist to measure the flow experience while using any web application, they are always customized to exactly fit their needs. Automated evaluation of flow with a software application would not be feasible because flow is a human experience which cannot be captured by software. Pace [13] already conducted research about flow and reasoned "a grounded theory of the flow experiences of web users". Starting from this basis and the empirical studies already performed it might be possible to develop a tool set for measuring and evaluating the flow experience while using a web application. This would be a useful tool to improve the evaluation of prototypes of web applications, too.

8 Suggestion of a new Form of Evaluation

A possible new form of evaluation is depicted in this section. After the investigation of the literature regarding the evaluation of prototypes of web applications up to date no literature regarding this idea was found. It is to be evaluated if this is a valid approach.

A possible approach to evaluate a web application regarding usability and layout can be the use of a modular web site during the prototype stage of the web site. A modular website with a layout proposed by the developers is provided as a prototype. Testers can move the elements around as they prefer. If an element is moved, this data is transmitted to the developers. It also has to be noted if elements are moved back to were they were and how often they were used in which position. This way, at the developers end, a database of the different layouts of the modular elements can be built. At some point in production then a snapshot of the database is made. The developers can now process the information about the layouts they have. The most commonly used layouts can be defined by frequency of occurrence. Users who wanted to just try moving elements around without using them can easily be recognized this way. If many users prefer a different layout than the standard layout, it has to be evaluated why this is the case. The web application can be improved according to the knowledge gathered or simply to the preferences of the majority of testers. This process can

be repeated iterative until the final version of the prototype. The final web application is then a frozen version of the last iteration.

Advantages of this system are that many users can test the web application at the same time. And it is possible to satisfy the majority of users. Disadvantages are that testers might miss to be able to customize the website to their needs if they prefer a different than the standard layout. The web application has to be set up up to transmit any movement of the modular elements. This functionality is not needed for the final version anymore.

9 Overview and Classification

For the evaluation of a web application the person processing the results always has to have a thorough understanding of basic web design principles. Only then he will be able to understand and interpret the findings of the evaluation. And only then he is able to improve a web application or its prototype. All tools and methods have this in common. But there are some features which only certain evaluation tools or methods demand. These features resemble constructs from the Repertory Grid Technique presented earlier.

- Evaluator: The person performing the evaluation can be an expert, a novice, an expert or a novice or it can be an automated test where software is running the evaluation.
- Resources: Special resources can be needed to conduct an evaluation of a web application: A Server, a proxy-server, special software, a checklist, etcetera.
- Special Knowledge: Some tools require other tools or special languages to be learned before they can be utilized properly.
- Access to Log Files: Some evaluation methods require access to log files. These can be produced client-side, server-side or at a proxy server.
- Location: It is possible that the evaluation of a web application can only be done in a certain location, for example a laboratory with special equipment.
- Type of evaluation: The evaluation can be formative and / or summative, or quantitative and / or qualitative.
- Content: The evaluation is restricted to formal content or also implicit information can be evaluated.

An overview over the all the different features claimed the different tools and methods can be consulted in table 1. All tools and methods to evaluate prototypes of web applications presented in this paper are merged into this one table for reference.

Table 1. Evalution Tools and Methods - An Overview

Feature	Option	1	2	3	4	5	6	7	8	9
Evaluator	Novice	x		x	x	x	x			
	Expert	x	x	x	x	x	x			x
	Software							x	x	
Resources	Checklist	x	x							
	Proxy Server				x	x				
	Software			x	x	x	x	x	x	
	Database						x	x		
Special Knowledge about Tools	Required			x	x	x	x	x	x	
Access to Log Files	Server-side			x						
	Client-side						x			
	Proxy-side				x	x				
Location	Home	x	x							x
	Field	x	x	x	x	x	x	x	x	x
	Remote	x	x		x	x		x	x	x
Type of Evaluation	Formative	x	x							x
	Summative	x	x	x	x	x	x	x	x	x
	Qualitative	x	x	x	x	x	x		x	x
	Quantitative	x	x		x	x		x	x	x
Content	Formal	x	x	x	x	x	x	x	x	x
	Implicit	x	x	x	x	x	x			x

Legend of Table 1	
1	Keevil Usability Index
2	Web Usability Index
3	Web Usage Mining with WUM
4	WebQuilt
5	UsaProxy
6	GINIS
7	Quantitative Evaluation of Web Sites
8	Automated Evaluation with Guideline Review
9	Repertory Grid Technique

10 Conclusion

In this paper, several approaches to evaluate prototypes of web applications were presented and suggestions for missing tools were made. The functionality of the tools and methods has been explained. Additionally, they were divided into different categories according to their natures. In the respective categories, the tools were refined over the time. They always tried to overcome the shortcomings of their predecessors.

In general it can be said that the evaluation with checklists and interview is the most flexible form of evaluation. Both types are strongly related to each other. They don't need special resources or special knowledge of software and they are not bound to a certain location. All types of evaluation can be conducted, all types of content, formal and implicit, can be evaluated. The only downside is that the Web Usability Index and the Repertory Grid Technique are supposed to be performed using expert evaluators for best results. The Keevil usability index can also be conducted with novice testers, but experts would deliver more appropriate results.

Behavior-based evaluation with logging does need special resources in the form of software or proxy-servers. Also, access to the log files has to be assured. Without the log files the evaluation cannot take place. But WebQuilt and UsaProxy are the best tools to evaluate remote testers. Evaluating a prototype while it is still worked on does not make sense, only summative evaluation can be used on the finished stages of the prototypes. Qualitative evaluations can be made with all these tools like GINIS or web usage mining with WUM, but only some are suited for quantitative evaluations of web applications, like WebQuilt or UsaProxy. To be able to evaluate the information of the logged data every tool needs a special software, so this can only be done on location. Formal and implicit content and information can be evaluated. The interpretation of informal content is prone to the interpretation of the person evaluating the log files, so is to be taken with a pinch of salt. The advantage of these approaches is that normal users can be used to evaluate a web application. This is even desirable in some cases, because expert users would not make the mistakes of novice users. Their mistakes would be unnoticed and so no improvement could take place.

Automated tools supporting evaluation are superior to humans in the evaluation of the correctness of formal content of web applications. They cannot evaluate implicit information of web pages though. There is special software needed to run these kinds of evaluation as there are no human evaluators involved. This postulates that the person running the evaluation has a thorough knowledge of the software tools. For the quantitative evaluation of web sites also a huge database is required to be able to store the information gathered which is needed for further evaluation. In addition, this does not allow for qualitative evaluation like the automated evaluation with Guideline Review.

There are no tools to measure and evaluate the flow experienced by user interacting with a web application yet. Taking Csikszentmihalyi theory of flow [4] into account, flow in general can be experienced by novices and experts alike. Software cannot compute flow as it is a state of the human mind. Flow is not suitable for formative evaluation because the flow experience results from the whole experience. Missing or not working parts of a prototype would interrupt the flow. The

evaluation would be qualitative and could also be employed quantitative. It would be possible to evaluate content formal as well as implicit using the flow construct. As there is no tool yet, no statement about possibly required resources or knowledge of tools can be made.

All in all it can be said that each tool presented has a special domain where it is superior to the other tools presented. Which tool is best used to evaluate a certain prototype of a web application depends on so many factors that is has to be separately evaluated for each purpose independently.

REFERENCES

[1] R. Atterer, M. Wnuk, and A. Schmidt. Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 203–212, New York, NY, USA, 2006. ACM.

[2] C. Bauer and A. Scharl. Quantitive evaluation of web site content and structure. *Internet Research: Electronic Networking Applications and Policy*, 10(1):31–44, 2000.

[3] B. Berendt and M. Spiliopoulou. Analysis of navigation behaviour in web sites integrating multiple information systems. *The VLDB Journal*, 9(1):56–75, 2000.

[4] M. Csikszentmihalyi. In *Beyond Boredom and Anxiety*, San Francisco, CA, USA, 1975. Jossey-Bass.

[5] C. Floyd. A systematic look at prototyping. In *Approaches to Prototyping*, pages 1–18, Berlin / Heidelberg, Germany, 1984. Springer.

[6] I. Harms, W. Schweibenz, and J. Strobel. Usability evaluation von webangeboten mit dem web usability index. In *Proceedings der 24. DGI-Online-Tagung 2002 - Content in Context*, pages 283–292, Frankfurt am Main, Germany, 2002. DGI. 283-292. visited 10.11.2009.

[7] J. I. Hong, J. Heer, S. Waterson, J. A. Landay, and J. A. L. Webquilt: A proxy-based approach to remote web usability testing. *ACM Transactions on Information Systems*, 19(3):263–285, 2001.

[8] B. Keevil. Measuring the usability index of your web site. In *SIGDOC '98: Proceedings of the 16th annual international conference on Computer documentation*, pages 271–277, New York, NY, USA, 1998. ACM.

[9] G. Kelly. In *The Psychology of Personal Constructs*, New York, USA, 1955 (reprinted by Routledge, 1991). Norton.

[10] M. Matera, F. Rizzo, and G. Carughi. Web usability: principles and evaluation methods. In *Web Engineering*, pages 143–180, Berlin / Heidelberg, Germany, 2006. Springer.

[11] T. P. Novak and D. L. Hoffman. Marketing in hypermedia computer-mediated environments: Conceptual foundations. *The Journal of Marketing*, 60(3):50–68, 1996.

[12] T. P. Novak and D. L. Hoffman. Measuring the flow experience among web users. 1997. visited 10.10.2009.

[13] S. Pace. A grounded theory of the flow experiences of web users. *International Journal of Human-Computer Studies*, 60(3):327–363, 2003.

[14] X. Roche. HTTrack homepage visited 09.12.2009.

[15] Y. X. Skadberg and J. R. Kimmel. Visitors' flow experience while browsing a web site: its measurement, contributing factors and consequences. *Computers in Human Behavior*, 20(3):403–422, 2004.

[16] M. Spiliopoulou. The laborious way from data mining to web log mining. *International Journal of Computer Systems Science and Engineering*, 14(2):113–125, 1999.

[17] M. Spiliopoulou. Web usage mining for web site evaluation. *Communications of the ACM*, 43(8):127–134, 2000.

[18] F. B. Tan and L. Tung. Exploring website evaluation criteria using the repertory grid technique: A web designers' perspective. *Second Annual Workshop on HCI Research in MIS, Seattle, WA, 65-9*, 2003.

[19] J. Vanderdonckt and A. Beirekdar. Automated web evaluation by guideline review. *Journal of Web Engineering*, 4(2):102–117, 2005.

[20] G. Velayathan and S. Yamada. Behavior based web page evaluation. In *Proceedings of the 16th international conference on World Wide Web*, pages 1317–1318, New York, NY, USA, 2007. ACM.

Haptic Icon Prototyping

Dario Soller

Abstract— The development of more complex technical devices demand new sensory channels to transmit increasing abstract informations. The main communication of Human Computer Interaction (HCI) essentially uses visual and auditory based channels and is often overloaded nowadays. With the aim to balance the computer generated information on the different human sensory channels and to possibly communicate additional informations, the prototyping techniques in the field of haptic feedback and the more complex form of haptic icons is going to be discussed in this paper. General terms and questions of haptics are exposed and followed by several detailed explanations of current haptic prototyping techniques especially concentrating on tactons. They are of major interest in this paper, because representations of abstract information with icons always showed easier and better cognitive performances, no matter if it is in the visual or auditory modality. Their different prototyping techniques are presented in-depth. In the end we address unsolved questions in Haptic Icon Prototyping (HIP) and complete this paper with an outlook on the future in previously unused potential haptic icon feedback technologies and their possible future prototyping techniques. This paper provides a summary of the current state of research in the field of haptic icon design. Details on kinesthetic or force feedback are not part of this paper.

Index Terms—Prototyping, Prototype, Design, Haptic, Tactile, Haptic Icon, Tacton, Hapticon, Icon, Tangible, Touch, User Interface

1 Introduction

Since the 1960s mouse, keyboard and monitor have been our gateway to the computer. But "tangible interfaces and ubiquitous computing technologies are changing the human relationship to computing technology, and designers must take this into account when creating products and services"[22]. Ivan Sutherland, a founding father of virtual reality, suggested that the "human kinesthetic sense is as yet another independent channel to the brain, a channel whose information is assimilated quite subconsciously"[51] .

The scientific research in the field of haptic feedback has already been around since the 1990s. But still the amount of new devices supporting haptics is still relatively small and most of the time not implemented to their full possible extend. One reason may be the lack of proper low-fi haptic prototyping techniques being involved in early design states in general [10]. Therefore possible prototyping techniques should be reviewed and compared, especially concentrating on the prototyping of haptic icons, which are hiding great potential implementation possibilities and improvements for HCI. "The goal, inspired by the common use of audio icons in desktop interfaces and mobile telephony [13], is to allow for the design and construction of specific and short abstract tactile messages that can easily be interpreted by users with minimal cognitive effort"[24]. For instance, "if the visual system is overloaded, you can provide object identification information haptically without adding significant cognitive load"[53].

First of all the major terms and expressions in the field of haptic icons are defined, followed by examples for potential fields of application as a delicacy, sensibilizing for the practical context of haptic icons. Then the historical development of publications in the field of haptic icons indicates the research activity in this topic. After that we analyze the affected group of people, designing haptic icon user interfaces and working with them in the future. Subsequently important parameters and their domains of definition give the basis for the actual approaches on HIP and how they could be classified into hi-fi and low-fi prototyping techniques. Out of these results, domain proposals for the different HIP techniques and the different design situations are given. In the end open questions in the HIP research are listed and discussed, as well as a short outlook for up to date unused potential haptic feedback technologies and their possible future prototyping. This paper tries to give detailed views on the present state of haptic icon research, with the concentration on HIP.

- *Dario Soller is studying Media Informatics at the University of Munich, Germany, E-mail: dario.soller@campus.lmu.de*

- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009/2010*

2 Overview

This section provides initiatory details on haptic icons, their practical contexts, the history of haptic icon research and informations about the group of people actually using haptic icon prototyping. The most important part definitely is the summary of technical parameters, which are of major relevance when designing hapticons.

2.1 Definitions

haptic: With haptic, one means everything concerning the sense of touch. In the context of this work, it describes the transmittance of computer generated information by haptic user interfaces and special devices to the human tactile sensory system. Possible parameters for the human skin and thereby for haptic prototyping are kinesthetic (forces on and motions of body parts), tactile (materials and surfaces), vibrating, temperature and pressure sensations [54] *(more details in 2.5 Technical Parameters of Haptic Icons)*. Furthermore one should know that "the human haptic system is made up of two subsystems, the motor sub-system and the sensory sub-system. There is a strong link between the two systems. Unlike the visual system, it is not only important what the sensory system detects, but what motions were used to gain that information"[51].

Force feedback and other kinesthetic events are felt predominantly with the motor sub-system and are left out of this study, because of the differences in examination principles and the different prototyping devices claimed by the two haptic sub-systems.

haptic icon: a jingle like symbol or piece of abstract information consumed via the sense of touch of the human tactile sensory system. Haptic icons do not only legitimate by complementing other modalities as visual icons or earcons, but accomplish abilities of being an independent sensory channel, on which complex information can be transmitted separately on the multiplicity of cutaneous mechanoreceptors [30]. "They are structured messages that can be used to communicate to users non-visually"[14]. "In the visual domain there is text and its counterpart the icon, the same is true in sound with synthetic speech and the earcon. In the tactile domain there is Braille but it has no iconic counterpart. Tactons fill this gap."[13] It is still state of the latest research work, how to iconify certian events *(more details in* **??** *Haptic Icon Design)*. Slight differences between hapticons and tactons are made by Enriquez and MacLean[26] as well as Brewster[13], who define hapticons as the simpler version of haptic feedback similar to visual icons and earcons. Hence tactons are

discribed as laconic representations of ”complex interface concepts, objects and actions”[13] like navigating through directories on a pc via the sense of touch.

2.2 Fields of Tacton-Application

There are many potential fields of application for haptic icons in common tasks with technical devices today. Many of the referenced authors talked about practical adoption examples for haptic icons, of which a few are presented in the following passage as a further entry.

Swindells imagined the indication of a station’s genre of music by an unique feeling or pulse at the haptic radio tunning knob [54]. Another already implemented haptic feedback knob was introduced by BMW’s iDrive with which one can ”access secondary vehicle functions such as audio and climate-control systems. It varies the knobs feel (via programmed compliance and damping) to create a range of detent sensations, with different sensations mapped to different control functions”[25].

Hoggan proposes a multimodal combination example, where messages are presented through an auditory signal and the urgency of the email is expressed by a rhythm with a rough texture to the left hand side of the user’s waist[31]. A progress bar widget, for example, could be displayed by two sets of tactile pulses. The closer the two pulses the nearer the download is to finish[13]. Or the set of transducers on a belt tells the progress of the download with a moving pulse starting on the front and moving around the body in a clockwise direction [13]. Brewster gives us some more application examples, like tactons signifying additional information to the type of building (shop, bank, office-block, etc.), type of shop (cloths, food, etc) and the pricebracket of the shop (budget, mid-range, expensive)[13] one is standing in front of. Such wearable devices are of great importance where screens are limited or in interfaces for blind people [14]. A further potential application for blind people is given by McDaniel, who presents a vibrotactile belt, which indicates the location of someone in front of the user. The relative direction is communicated by the position on the vibrotactile belt, while the duration of the vibration displays the interpersonal distance [42].

Also more complex informations could be mapped into haptic icons. ”For example, with multidimensional data one dimension might be mapped to the frequency of a pulse in a tacton, another might map to rhythm and another to body location” [13]. Even a whole file system is representable by transformational tactons *(see* **??** *Haptic Icon Design)*. ”The file type could be represented by rhythm, size by frequency, and creation date by body location. Each file type would be mapped to an unique rhythm”[13].

2.3 History of Haptic Icon Research

The mass of information in the world wide web only provides a rare amount of information about the field of haptic icons and one soon finds out about a handful of authors and scientist which bear the brunt. The first Wikipedia articles concerning haptics appeared around 2003, but there are still no own articles for haptic icon or tacton, not to mention special prototyping techniques for this topic. The words ’haptic’ or ’tactile’ are not even alluded once in the article about prototyping on Wikipedia so far. An attempt with Google Trend was only available for ’haptic’ and hence had not been meaningful enough.

The number of publications *(see figure 1)* were investigated with the ’Google Scholar’ search engine. Searching for different expressions for haptic icon showed following results. The expression ’tacton’ also has a meaning in specialized medicine and had therefore unfortunately not been regarded furthermore. Starting in 1993 the topic came up and showed a slight increase of interest with little variabilities since then. Also years with no publications occurred during 1995-1996 as well as between 1999-2000. For the last four years the annually publications are nearly consistent at around 16-20 publications per year. One peak definitely is the year 2006 with 24 publications. The one paper addressing HIP directly in 2006, is the often refered to work ’The Role of Prototyping Tools for Haptic Behavior Design’ by Collin Swindells et al.[54].

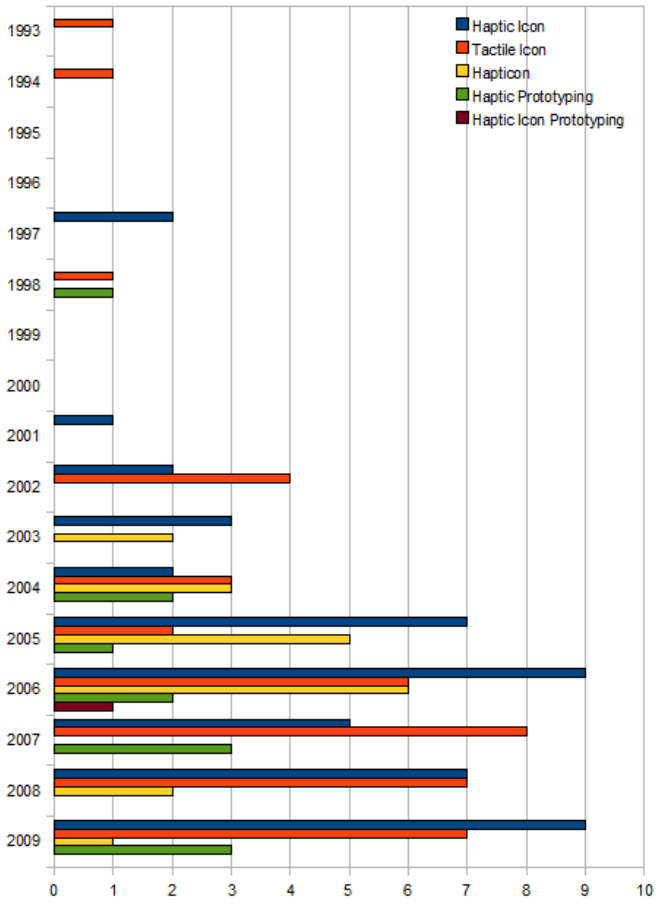


Fig. 1. Number of Publictions on relevant Key Words since 1993.

Again it is said, that these results are based on the limited expressiveness of the Google Scholar entries. But over all there are relatively few scientific works in the field of haptic icons and barely any concerning HIP invariably. It will be very interesting how these numbers develop in the near future, maybe indicating how overloaded visual and auditory based user interfaces effectively are today.

2.4 Affected Group of People

During the design of prototyping techniques, the affected group of people going to design haptic icons need to be familiar. This is necessary to really improve and find productive prototyping methods.

Programmers and Engineers: The first group consists of the engineers and programmers developing interfaces for technical devices in their laboratories. They have the knowledge base to handle ”more complicated user interfaces and program code to create sophisticated and/or novel haptic effects”[54]. On the other side they demand for great flexibility and control of the haptic icon prototyping technique. In the past most of the haptic interface developments ”has taken place in robotics or engineering labs, concentrating on the challenges inherent in building low cost, high-resolution devices with realistic size, power and safety performance”[13]. Somehow only a little amount of realized haptic feedback implementations and an approved usability of haptic feedback arose from this research [13]. This is about to change with the adoption of certain haptic prototyping techniques by the design community [22].

Designers: Designers are less interested in programming details and low level hardware configurations, but with this trade-off of flexibility they gain fast iterations of prototypes concentrating on higher level concepts like usability of haptic icon feedback [54]. With more advanced tools and techniques supporting the fast design processes,

one can expect more mature innovative haptic interfaces [22]. "Devices are now available that allow the use of tactile displays so the time is right to think about how they might be used to improve interaction."[13]

Users: This group is willing to learn new interaction concepts, for example selecting between pre-defined components in order to be supported in their actions at best. They are satisfied with "stock customizations that are specialized to their particular device"[54]. Selecting a vibrotactile ring tone on a cell phone GUI would be an example here.

2.5 Technical Parameters of Haptic Icons

Beginning to design haptic icons one should pay downright attention to gain detailed knowledge about the physical characteristics of the skin. Technical parameters, especially their expedient ranges and the mutual manipulation of parameters are still not completely understood and subject of current research. Due to the temporal and spatial acuity, the skin could be the key medium to complement visual and auditory elements. "Visual icons can convey complex information in a very small amount of screen space, much smaller than for a textual description. Earcons convey information in a small amount of time as compared to synthetic speech. Tactons can convey information in a smaller amount of space and time than Braille"[13]. As a result of that hatpic icon designers are refused to draw directly from the field of music theory. But still the similarities between audio and tactile signals, like the way they are technically produced for example, seem to give little initial aid in the design of hatpic icons [17].

Fig. 2. Different Mechanoreceptors in the Skin [1]

On a closer view on the actual structure of the skin's mechanoreceptors *(see figure 2)* some limitation already become clear. Hale and Stanney[30] give a good overview about the different properties of the human skin:

- the skin's sensitivity depends on its size (large receptors have poor spatial resolution)
- density (many receptors in a given area results in high spatial acuity)
- frequency range (receptors don't perceive signals outside their range)
- nerve fiber branching (higher branching leads to spatial and temporal summation of signals)
- the type of stimulation (skin motion or sustained pressure) affects the degree to which individual mechanoreceptors are activated

In the following line-up you will find, different parameters, which had been elaborated in the different investigated works:

Duration: A certain number of meanings could easily be encoded in pulses of different durations alone [15]. Investigations by Gunther [29] show that vibrotactile "stimuli lasting less than 0.1 seconds were perceived as taps or jabs whereas stimuli of longer duration, when combined with gradual attacks and decays, may be perceived as smoothly flowing tactile phrases"[13]. Taps are experienced by sudden attacks, while a gradual attack is perceived as a rising pressure on the skin [13]. But in any case, stimuli must be at least 5.5ms apart to ensure that the cutaneous signals are perceived individually [50].

Amplitude: The amplitude, pressure or intensity of a tactile stimuli is proposed by several scientist ([13] [17] [53]) as an useful parameter in tacton design, but only if certain thresholds are factored into the haptic icon design process. Gunther writes that the intensity range is felt from the level of detection till approximately 55dB. An intensity above 55dB is perceived as painful [28]. Pressure sensations are activated by forces greater than 0.06 to 0.2 N per cm [50]. As a further limitation this range has no linear or homogeneous resolution, as it deteriorates at a level of 28dB [49], which is proposed by Brewster to be a useful maximum level of intensity [13]. Although Gunther found out that the just noticeable difference (JND) in detecting intensities is carried out in steps from 0.4dB-3.2dB [29], it is suggested by Gill not to use more than four different amplitudes within a single context [27]. Sherrick describes the JND as a "values range from 5 milligrams on a woman's face to 355 mg on a man's big toe"[50].

More sophisticated shapes of designing with amplitudes had been confirmed by Brown et al. [17]. They introduced dynamic transitions of amplitudes, forming linear, exponential and logarithmic in- and decreases of vibrations. Worth mentioning is that the recognition rates of logarithmic in- and decreases showed weaker results than the other ones.

The most suitable approach of designing with the amplitude parameter is certainly reported by Brown et al.[15], as they suggest to leave the control of actual intensities in the hand of the user. Having the possibility to customize the amplitudes of the tactile signals in a predefined range individually, will ensure the best general performances in terms of usability at the same time.

There seems to be an intense dependency of amplitudes and frequency parameters used in haptic icons, so that several researchers ([6] [53]) have claimed "to combine [them] into a single parameter to simplify design"[14].

Frequency: As widely known the perceivable range of the human ear is from 20Hz-20kHz, but "the practical frequency range of the skin is much smaller, ranging from 10Hz-400Hz"[21], being most sensitive at 250Hz [15] [28]. Brewster even writes of an tactile spectrum of 20Hz-1000Hz [14], but most of the transducers, actuators and vibrotactile devices even have a much more restricted range than 10Hz-400Hz [15]. Experiments by Tan showed that users could distinguish three categories of frequencies between DC and 300Hz [35]. Frequencies below 100Hz are experienced as periodicity or buzzing whereas higher frequencies are felt smoother or diffuse [58]. For example Hoggan and Brewster did successfully use frequencies of "6 Hz (slow motion, very rough), 70Hz (fluttering slightly faster motion, rough), and 250 Hz (smooth)" in their experiment [32]. Another discrepancy to the abilities of the human ear, is the identification of absolute frequencies. "Making relative comparisons between stimuli is much easier than absolute identification"[13]. Therefore a maximum of nine different frequencies is suggested by Gill [27]. A similarity to the field of sound is that "a change in amplitude leads to a change in the perception of frequency so this has an impact on the use of frequency as a cue"[13]. As a consequence of this strong affection of the different parameters, they all have to be taken into account simultaneously. This then leads us to specifications like that of Biggs: "Vibration from a single probe must exceed 28 decibels (relative to a 1ms peak) for 0.4- to 3-Hz frequencies for humans to perceive."[9]

Waveforms: With waveforms one means a sine wave whose amplitude is modulated by a second frequency different to the first one [15]. Users are even able to differentiate between sine and square waves [13], but more subtle differences are critical to perceive clearly [29]. The auditory equivalent for 'waveform' is 'timbre', which "is a key attribute in earcon design"[11]. This makes waveforms the important parameter in crossmodal combinations or substitutions [34] as well as an important texture design parameter for haptic icons. "So, in tacton parameter design, waveform can be correlated to the 'texture' of tactile stimuli"[32]. Furthermore the often referred ([15] [16] [46]) to tactile impression of 'roughness' (with a recognition rate of 80% [15]) can be achieved with waveforms. The recognition rates of waveforms are with 94% over all very promising, as it seems that waveforms combine the advantages of the amplitude (recognition rate of 61%) and frequency (recognition rate of 81%) parameters [32]. In code of practice 250Hz sinusoids modulated by 50Hz or 30Hz frequencies have been proven positive application [15] [32]. But "since tactons should be as short as possible in order to communicate information quickly, it is important to choose higher modulation frequencies, so that shorter pulses can be used"[15].

Rhythm: A combination of pulses with different duration forms first basic rhythmic units [15] [13]. Additionally variations in tempo [14] in which these rhythms are played, can provide tactile 'melodies' [56] and further design possibilities. Brown et al.[15] examined recognition rates for rhythm in tactons of 93% which is similar to that of melody in earcons (90%) [43]. In an experiment using "temporal patterns (rhythm) along with frequency and amplitude to encode speech information in vibrations, [Summers[52] found out] that participants mainly used the information obtained from the temporal patterns, rather than from the frequency/amplitude modulation"[15]. "The amodal attributes between our senses of hearing and touch include intensity, rate, rhythmic structure and spatial location"[39]. All this gives further evidence that it is possible to draw from the field of music and that rhythm is a major parameter in tacton design [17]. Even though this association should always be carefully observed.

Swerdfeger et al.[53] elaborated detailed results for the use of rhythmic elements in the design process of haptic icons. They report that: "The most groupable stimulus characteristics are note density and rhythm, as long as the rhythms are not syncopated." Further results of rhythmic stimuli, which dominated the users' perception are:

- Rhythmic differences between melodies dominate other distinctions. And the rhythm of a stimulus is perceived rather than its onset.
- Perceived quantity of notes is a major grouping factor. Hence replacing a quarter note with two eighth notes can increase expressiveness while maintaining groupability. Groups of rapid (eighth) notes are perceptually salient in rhythms.
- Items that only differ in phase are grouped together. While distinct groups that contained eighth notes were often confused by participants.

In another conclusion by Swerdfeger et al. [53], they found out that "low amplitude sustained notes surrounded by staccato notes are often grouped with those that have rests in the same position as the sustained notes". So the more complex tactile rhythms get, the harder mutual differences can be identified, rising confusion by users. This underlines that simple haptic stimuli should be aspired [48].

Body Location: The human skin embraces a surface of 1.5-2m^2, an expanse which had been barely used for information display yet. But the transmittance of information with spatially allocated transducer has been successfully proofed in different researches [57], so we will take a closer look on present facts. The spatial acuity and sensitivity of the skin differs on different body locations. For instance, the two-point threshold shifts smaller from palm to fingertips, with a spatial resolution of 2.5mm on the index fingertip [50]. Haptic interface designers must consider this as well that distributed tactile stimuli get groupable by their regional closeness. Often haptic feedback concentrates on the hands and fingers because of the high sensitivity and resolution ability [49]. However, under real conditions the hands are mostly used for other tasks [13], that's why Sherrick et al.[49] suggest to use other suitable positions like back, thigh and abdomen. But vibrotactile transducers should never be placed near the head, as they influence the ear, producing undesirable auditory side effects [28]. If haptic icons are learned on one location they can be transferred to other locations nearly without any loss in recognition [49]. Already with "an 3x3 array of stimulators located on [the users] back, lines and geometric shapes can be drawn"[13]. This operates like tactile, geometric animations, which can vary in time and location that's also why they are called spatiotemporal patterns [13]. Nevertheless the most elaborated use of body locations, was done by Rupert [47], harnessing the complete torso. 128 transducer indicated pilots the relative position of other objects (e.g. airplanes, borders, horizon) in a 3D space around them.

Temperature: Due to the latencies of temperature sensations, hardly any works had been found referencing temperature as a parameter in haptic icon design. Anyhow some thermal-tactile facts still might be useful. The skin contains cold-sensitive and warm-sensitive thermoreceptors, which detect temperatures of 5-43 $^\circ$C and 30-48 $^\circ$C. The pain threshold is beneath 17 $^\circ$C and over 44 $^\circ$C. Remarkable might also be that cold-sensitive thermoreceptors exist ten times more often than warm-sensitive ones. Thermoreceptors occur with an irregular disposition, depending on the body location. A fast change from cold to hot can cause a paradoxical response to heat, which is a thermal shock, based on a sudden discharge of the cold-sensitive thermoreceptors.[38]

Finally rounding off all these technical attributes, Hoggan notes that "tactile feedback becomes ineffective at vibration levels of 9.18 9.45 g/s and above suggesting that audio feedback should be used at these levels"[33]. Furthermore, Tan actually measured bit rates for perceiving vibrotactile stimuli of 2-3 bits per second [35].

Maybe in the past designers were just astonished by the sheer amount of affected parameters, that they have often avoided to max out the human tactile sensory channel to its full potential. The difficulty of building prototypes covering all these parameters and the difficulty to integrate them into an early design status prototype, can be outlined as the technical novices' difficulty [22]. In the next sections we will see ways of fast and efficient haptic icon prototyping, constituting on the presented technical parameters.

3 Haptic Icon Prototyping

The availability of consumer products are inevitably bound to the conjunction of sensors and actuators in devices featuring haptic feedback. "Sensing technology in manufactured products is efficacious and affordable yet requires the input of trained engineers to specify and prototype. So, product and interface designers often don't have the opportunity to explore this technology's potential benefits early enough to incorporate it in a products design"[22]. Before specifying a production-ready haptic interface, designers need a possibility to experience these interactions in order to truly imagine the users necessities. All this pictures the demand of adequate prototyping techniques, especially in the field of the more complex form of haptic icons.

Recent trends in the market of mobile devices show that a lot more "tactile actuators with increased degrees-of-freedom" have been integrated [24]. In the following section a selection of different useful actuator and transducer components will be presented. Enjoy!

3.1 Haptic Devices

There are almost as many technological devices as affected parameters *(see 2.5 Technical Parameters of Haptic Icons)*. Most of the time a plate or an array of pins in direct contact with the skin is used. The *Tactaid VBW32* transducer *C2 Tactor (see figure 3 left)* for example, is an inertial transducer, in which a mass is suspended on a spring, being

all placed into a rigid case. It is produced by Audiological Engineering Corporation [7]. When it receives an alternating electromagnetic signal, "the user feels the vibrations through the case itself"[17]. Many experiments had been made using this actuator e.g. being attached to a finger tip.

On the other hand there is the *C2 Tactor (see figure 3 right)*. It is a voice coil transducer with an explicit contact point through which the generated vibrations are transmitted to the user. It is produced by Engineering Acoustics Inc. [23]. Nearly as many experiments were done with the Tactaid or the C2 Tactor or even both at once. Both can be assessed as the most established tactile prototyping components yet. Even though Brown et al.[15] indicate that the C2 Tactor seems to be more precise than the Tactaid, as results have been more consistent with the C2 Tactor.

Another example would be the *Pin Arrays* produced inter alia by Summers et al.[36], which generally hold great potential in displaying "very fine cues for surface texture, edges, lines, etc."[13]. The often mentioned *VirTouch tactile Mouse* seems to be vanished into thin air, as no serious source can be found, except what Brewster [13] wrote. Furthermore Brewster alludes another example, which was ceased, the *Optacon* by TeleSensory Inc. [2]. But other pin array devices like the dynamic Braille cells are still available [3] with up to 80 pins. These devices could be diverted from its intended use of presenting Braille text to display haptic icons in a prototyping context [13].

Vibrotactile belts were used for several experiments by Cholewiak et al.[45] and lately by McDaniel [55][42]. These belts are very often just assembled with C2 or Tactaid transducers. Here another example set up used by McDaniel et al.: "The belt consists of 7 tactors equidistantly placed in a semi-circle with the first, fourth and seventh tactor at the users left side, navel, and right side, respectively. Each tactor consists of a pancake motor of diameter 10mm and length of 3.4mm, and operates at 170Hz"[42].

Lately some more extensive haptic devices appeared on the consumer market like *Mobile Phones*, supporting "vibration patterns, which accompany midi ringtones"[18]. Especially Motorola [20] and Nokia are on the forefront of these innovators. But ringtones and games in mobile phones are enriched also by companies like Immersion [37] with their *VibeTonz* technology. Immersion even came up with a *CyberTouch Glove* [13]. On the other hand Nokia invented *Digital Pens* indicating battery status and other alerts with vibration patterns [4]. Some of these devices could boost the future design of haptic icons immensely.

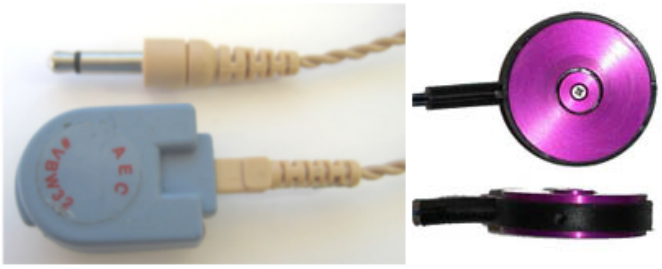


Fig. 3. The *Tactaid* [17] and the *C2 Tactor* Transducer [23]

3.2 Haptic Icon Design

Enriquez et al.[25] did important spadework in the theory of haptic icons. They examined for example the internal structures of tactile stimuli and haptic icons. Similar to speaking, phonemes can be namely defined in haptics. The so called *haptic phonemes* form the smallest distinguishable building blocks and are therefore the smallest recombinant module of a physical haptic stimulus to design with. Subsequently Enriquez et al.[25] talk about two combination approaches *(see Figure 4)*:

- Concatenation: Phonemes are combined serially to create a word, following an analogy with English word construction.
- Superposition: Phonemes are combined in parallel to create a word of the same length as the longest original phoneme, following a musical chord analogy.

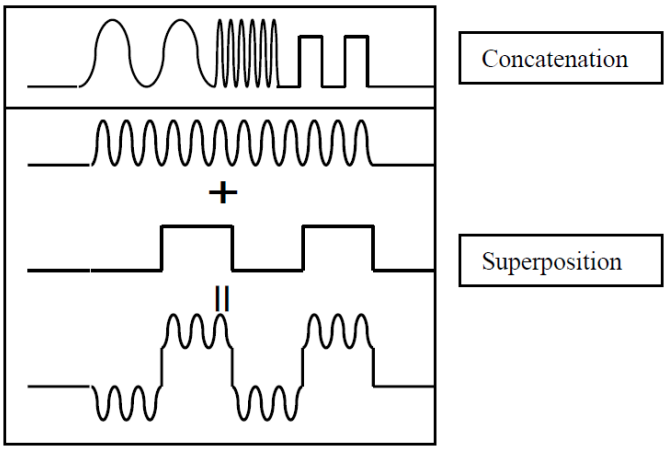


Fig. 4. Haptic Phonemes - Smallest Building Unit for Haptic Icons [25]

Haptic phonemes are constructed of simple waveforms, specifing the temporal path of the signal and a frequency, specifing the rate at which the path is traversed in a certain amplitude. "These phonemes can be assigned meanings which, when combined to create *haptic words*, can represent increasingly elaborate families of concepts that are related both semantically and haptically"[25]. They should meet the correlating requirements of being differentiable, easy to learn and easy to remember.

A further classification was conducted by Brewster and Brown [13] [14] devising tactons in three groups. First of all there are the *compound tactons*, which mean the serial combination of small tacton blocks, composing little messages. This is certainly equivalent to the concatenation of haptic phonemes by Enriquez et al.[25] mentioned before. Some tasks in a file system represented by compound tactons should give a better illustration of how this works. "The mapping is abstract; there is no intuitive link between what the user feels and what it represents"[14]:

- create: increasing high frequency pulse
- delete: decreasing lower frequency pulse
- file: two falling notes
- directory: two rising notes

Thereby the compound message 'create file' or 'delete directory' can be expressed by simply combining the haptic meanings.

This ties in with the representation of file type, creation date and file size, which could be displayed by so called *transformational tactons* introduced by Brewster and Brown in 2004 [14].

- file type: indicated by a certain rhythm
- size: smaller files by high frequencies; bigger files by lower frequencies
- creation date: certain body location tells a relative time

"If two files were of different types but the same size they would be represented by different rhythms with the same frequency"[14]. Further examples would be the already mentioned progress bar widgets *(see 2.2 Fields of Tacton-Application)*.

Last but not least there is the type called *hierarchical tactons* [13], capable of presenting hierarchical data structures. In computer science

tree structures are an important and common concept in data structures. It stands to reason that hierarchical tactons can be very useful in certain situations of HCI and HIP. Here a possible tactile encoding by Brewster and Brown [13]:

- Level 1: top tree node is a basic rhythm using a sine wave (family tacton)
- Level 2: compound tacton build out of the family tacton and adding to it e.g. as a square wave in a higher frequency
- Level 3: the tempo of the compound tacton from level 2 is changed

In order to ideally enable a user to complete his/her desired task, a haptic icon must [54]:

- Function technically (be physiologically perceivable, and relate to the users preconceived mental models of the task)
- Function socially (fitting into the tasks social and cultural milieu)

3.3 Prototyping Classifications

In this section we want to take a closer look on a selection of haptic icon prototyping examples. Due to the complex nature of haptic icons a distinction in horizontal and vertical prototyping seems not very reasonable. Horizontal prototypes would have to demonstrate the whole spectrum of haptic icon features of an interface, without actually implementing them, which would mean that one can not even feel the hapticon? This controversy may only be useful in combination with wizard of oz approaches, which are seperately discussed *(see 3.3.1 Low Fidelity Techniques)*. On the other side vertical prototypes don't make sense as well, because as soon as a working haptic feedback technology is integrated into a prototype, it should be no big deal to play and test different tactons with it, which brings us more into the direction of hi-fi prototyping at once.

Latest technological developments are promising an easier prototyping in the near future. Cottam and Wray note that "for designers to sketch tangible interfaces, their physical-computing toolset should give them a way to connect a GUI to sensing technology or actuators, preferably with some control over the translation between user action and interface event.[...] The entry barrier has been slowly lowered through the development of higher-level protocols and devices such as microcontrollers or A/D converters for hobbyists, artists, and designers"[22]. One of the first successful protocols for controlling hardware was MIDI in 1983. "Toward the end of the 1990s, other microcontrollers and physical-computing toolkits entered the market, including Easy IO, Teleo, Phidgets, and Arduino"[22].

According to Swindells et al.[54] an ideal set of HIP tools should support:

- All haptic types (kinesthetic & tactile) and affected parameters
- Interaction between all degrees-of-freedom (horizontal) and many levels-of-detail (vertical)
- Rapid iterations between various static & dynamic behaviors and reactions
- Completely representing the psychophysical capabilities of the user (ergonomics) via a standard set of mathematical relations
- Having easy-to-understand mental mappings between the underlying mathematical representations, the interaction widgets, and the final haptic renderings
- Providing usable interaction widgets for designers to effectively create and modify haptic renderings
- Integrating seamlessly with other haptic development tools, and development tools for other sensory modalities (vision, hearing, smell and taste)

"A designer rarely has a clear idea of the perfect, finished haptic behavior before starting the design [and] needs to be able to create several possible behaviors, [being] able to rapidly compare these behaviors"[54]. Hi-fi prototypes typically offer higher fidelity and better contextualization support, than low-fi prototyping who offer more flexible brainstorming options. But depending on the stage of an evaluated project, the right kind of prototype has to be defined, which is called 'appropriate prototyping' [19]. "There can be a tendency to try to build the most complicated prototype possible where a much simpler one will actually better communicate the design through the maintenance of ambiguity"[59]. Apparently there is a great need for fast iterating, low-fi prototyping techniques in haptic interface design, that's why we straight head for it now.

3.3.1 Low Fidelity Techniques

In HCI paper protoyting is a well established low-fi prototyping method for visual based interfaces. However, an effective prototyping substitution for the field of haptics is missing. Especially if haptic stimuli are used as a major information sources [19]. As we already now *(from section 2.1 Definitions)*, haptic icons do communicate abstract informations, which obviously stands in direct contradiction to the limited expressiveness of low-fi prototypes.

Nearly all referenced works speak about the importance of early impressions and free (simple, inexpensive) explorations in the design process of haptic interfaces. "As polished and realistic as a virtual rendering of a physical product might be, it can never fully express or explore the designs most important aspect: How the products users interact with it? How do they hold it in their hand? How does it rest if its lying on a table? How do they pick it up? A quick physical sketch of a product, however, is quite useful for the design process"[22]. When the designers' imagination for useful features meet ergonomics and behaviors in daily routines and surroundings, it is this inter-coordinated experience of situations and requirements, which really originates creative solutions for enhanced user support and improved usability in the end. "The ability to produce early prototypes is needed not only for the exploration and evolution of the design space, but also for making it possible to build the necessary bridge between users and design early in the process"[19].

Fig. 5. Low-fi Sketching Materials used at the NordiCHI Workshop, Low-fi Haptic Prototyping Example, Low-fi Wizard of Oz Example [12]

Low-fi prototypes "are sketches because they can be rapidly created and iterated [serving] only as useful approximations of the

final product. They are accurate where they need to be to inform the design, and are 'duct-taped' everywhere else"[22]. Right from the outset low-fi prototyping implies these aspects of experience, being uninfluenced of theoretical concepts, which often pinpoints fresh approaches to a problem. Thus the granted relevance for low-fi haptic prototyping seems justified, so let's discuss a few ways of them in the ongoing section.

Results of the *NordiCHI 2008 Workshop* [19]: A group of 16 people from academia and corporations like Nokia participated with the aim to share their knowledge in low-fi haptic prototyping. "The success of the Wii and the iPhone together with the development of low-cost force feedback devices have put haptic feedback within the scope of the individual user. Hence, there is an increased need to develop heuristics, guidelines and standards for its use"[19]. The participants playfully got their hands on rapid prototypes, made of everyday materials *(see figure 5 upper left)*. Therefore a big selection of materials purchased in ordinary department, hobby and toy shops was available to them. Still during the workshop some things emerged to be missing and give good advice for future strategies of low-fi haptic prototyping. The missing materials have been fast glue, velcro, more particles, stronger metal wires, more sounding stuff, small active things like vibrators or sound sources, everyday objects, materials that behave in weird ways and second hand shop things like cheap electric machines etc.. Strong magnets especially turned out to be a very convenient item to play out rapid prototypes.

The workshop spawned haptic prototypes like the 'walk with the wind navigator', the iBall for throwing music, emotional interfaces (mobile and stationary), a navigation ball, an eyes free music player and haptic breadcrumbs, which unfortunately all did not receive a separate explanation. But their main conclusion of this workshop certainly is that low-fi haptic prototyping needs sophisticated scenarios to gain a qualitative meaningful contribution to the design process. The following guidelines sum their experiences made during the workshop:

- Have a clear idea about the goals of your prototyping and think about what kind of responses you want (the level of polish changes the feedback)
- Use as many materials as possible and build up a wide collection of materials
- Get together a good mix of people
- Put effort into making good scenarios
- Use wizard of Oz to prototype advanced functionality
- Have fun!

Some more remarks on the use of Wizard of Oz for haptic icon prototyping *(see figure 5 bottom)*. The texture of haptic icons are of subtle nature, dependent of a big variety of attributes. The ranges in which humans are able to perceive them had been discussed before *(see 2.5 Technical Parameters of Haptic Icons)*, but this not immediately means that these parameters can be created by humans within a Wizard of Oz experiment. Furthermore even rough approximations of haptic icons can probably not be recreated 100% in different test cases. This once again clarifies the limits of low-fi prototyping, which have to be considered with well-informed caution. Hence, low-fi haptic prototyping with Wizard of Oz better supports horizontal prototyping approaches than vertical prototyping, as it is in fact possible to represent a great variety of haptic icons, but only in a rough guise.

Even with sketching materials used in the 'NordiCHI 2008 Workshop' it stays difficult to mimick realistic digital reactions [22]. But without appropriate sketching materials for haptic interaction, many design issues "will be experienced for the first time only after a costly prototyping phase requiring engineer input and the creation of a prototype built with nondisposable materials that are difficult to use iteratively.[...] Designers should be able to work with physical-computing materials quickly, iteratively, and as fluidly as possible."[22]. Some approaches coming close to solve these demands, will be presented in the next section.

3.3.2 High Fidelity Techniques

In this section hi-fi prototyping examples will be viewed, even though some of them also claim to be the equivalent to the low-fi technique 'paper prototyping' [54], they still all use complex hardware set ups and elaborated software and that's why they were classified as hi-fi prototyping techniques within this work. Positive aspects of hi-fi prototypes are an extended degree-of-freedom, supporting horizontal and vertical design issues as well as different levels-of-detail for concept visualization and mental models of the planned device. This results in pretty realistic user impressions at the expense of financial investments in specialized sketching technology or the cost of many hi-fi prototypes.

Tactile Handheld Miniature Bimodal (THMB)[44]: The THMB is held with the left hand, having the thumb located at the upper left side of the device. A small visual display kindly reminds of how a PDA is constructed. "The tactile display consists of a stack of eight piezoelectric benders *(see Figure 6 upper right)* intercalated between brass rods, which protrudes slightly through a narrow slit." This array of piezoelectric actuators arouses the thumb tip's skin by bending. The amplitude of a piezoelectric actuator depends on the voltage applied across its electrodes, which is controlled by a PC host running Linux. Intermediate electronics filter and amplify the control signals, generating control voltages from +/-50V with a sample rate of 3125 samples per second. "They are encoded with a single byte and therefore can only take 256 different values." With the THMB miscellaneous tactons can be induced and examined in conjunction of a visual modality.

BubbleWrap: a textile-based electromagnetic haptic display [8]: A very recent example is the BubbleWrap *(see Figure 6 bottom left)*, "a matrix of electromagnetic actuators, enclosed in fabric, with individually controllable cells that expand and contract." Information is either display by vibrations or by shape and firmness. Unfortunately there is no information about the back-end and the control system of the BubbleWrap. But this might be still a very useful prototyping tool for haptic stimuli in wearable devices or an armrest of a chair.

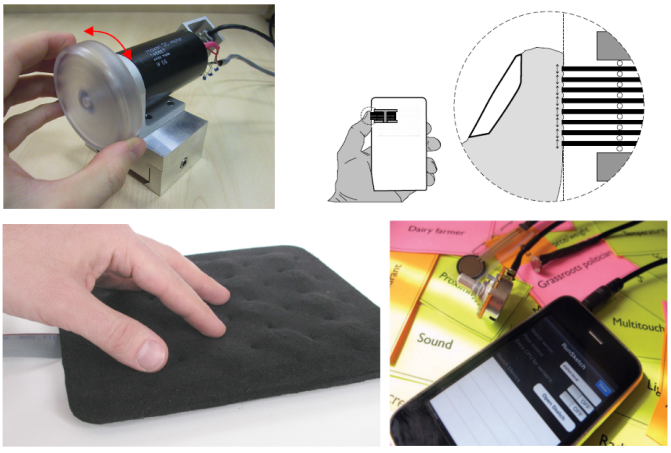


Fig. 6. The Haptic Knob [54], the THMB device and a close up of its tactile display [44], the BubbleWrap [8] and the NADA [22]

Haptic Knob with the Haptic Icon Editor Software [54]: Swindells et al. software follows a tile concept, in which one can design haptic icons via drag&drop. It is divided into three interaction regions: a waveform editor, a tile palette (position, velocity, acceleration tiles) and a tile pane. They accentuate that "organizing and editing collections of haptic tiles into more sophisticated haptic behaviors

is important because prototyping is typically an iterative learning process." Based on a set of mathematical equations, they actually calculated spatial and temporal representations for haptic icons, which their haptic knob *(see Figure 6 upper left)* is able to display. "The knob operates with an update rate of 10kHz, 0.001 ° positional accuracy, and 180 mNm maximum continuous torque" and is controlled by a "custom real-time platform middleware infrastructure", which again is simply a real-time Linux PC attached to an I/O board. The best thing about this workflow is that a haptic icon designer gets appropriate and fast mental mappings, as all template like haptic modules or tiles can immediately be felt and visually viewed. The limitation is the single degree-of-freedom of this prototyping tool.

Sketchstools Network Analog and Digital Adapter (NADA)[22]: This is an open source hardware and software toolkit consisting of a Java application, a microcontroller (A/D converter) and a custom circuit board, integrating additional sensors (heat, motion, force, etc.) and actuators (light, motors, fans, etc.). The innovation lies in the possibility to simply connect these sensor and actuator components directly into the headphone jack of an iPhone or iPod interacting with e.g. a Flash application *(see Figure 6 bottom right)*. "Our goal in developing these tools isn't to extend iPhone or iPod functionality. These devices simply happen to be well-qualified bases for our sketching platform because of their multitouch capabilities, software development kit, and large high-resolution display (relative to that of other mobile devices)." Their intention is vividly exemplified by this slogan: "We'll continue to strive to create tools that make sketching in hardware as simple as putting pen to paper." And indeed this really seems to be a powerful way of fast and iterative prototyping, incorporating a wide range of HCIs through the compatible selection of sensors and actuators, thereunder also ones supporting haptic feedback.

3.4 Haptic Prototyping Data Analyses

During prototyping most of the times a huge amount of data about the user performance is recorded. In this section useful analyzing methods for HIP data are disclosed briefly.

Analysis of Variance (ANOVA): This is a collection of statistical models, testing whether the means of several groups of a variable are all equal. This informs us about significant differences of user studies and therefore leads to error rates of a test case. Hoggan and Brewster for example used a "standard two tailed one factor ANOVA analysis, based on the critical values of the F distribution, with alpha=0.05"[32]. Also Brown et al.[15] [18] used ANOVA analysis to successfully proof their hypotheses. If there are more variables, what often can be the case in HIP due to the amount of affected parameters, it is maybe more suitable to use a specialized version of the ANOVA called *Multivariate Analysis of Variance (MANOVA)*. McDaniel et al.[42] demonstrate the handling of ANOVAs. "The reported ANAOVA results are from a two-way ANOVA on complete tacton recognition accuracy through location and rhythm".

Tukey's Honestly Significant Difference (HSD) Test: This is a statistical test in the form of a single-step multiple comparison, also matching all possible pairs of means. That's why the Tukey's HSD Test is often used post hoc in conjunction with previous ANOVA analysis, emphasizing or confuting each others' results [17]. For example did "post hoc Tukey HSD tests not show any differences in the individual pairs" of an experiment by Brown et al.[17].

Multidimension Scaling (MDS) Plot: A MDS plot is capable to sort common clusters in data structures, which can be visualized most of the times in 2D or 3D graphs. A pairwise comparison in a given dissimilarity matrix assigns locations for each item in a N-dimensional space *(see figure 7)*. Thereby complex large dimensional matrices are reduced to N-dimensions, maximizing the variance of the examined data. For instance, this can show "how users perceptually organize a set of stimuli"[53]. Enriquez et al.[24] used "2D multidimensional scaling plots to quantitatively show perceptual differences between several haptic icons". Hollins et al.[41] even succeeded to determine "dimensions such as hard/soft and slippery/sticky for real tactile surface textures"[53] in MDS plots.

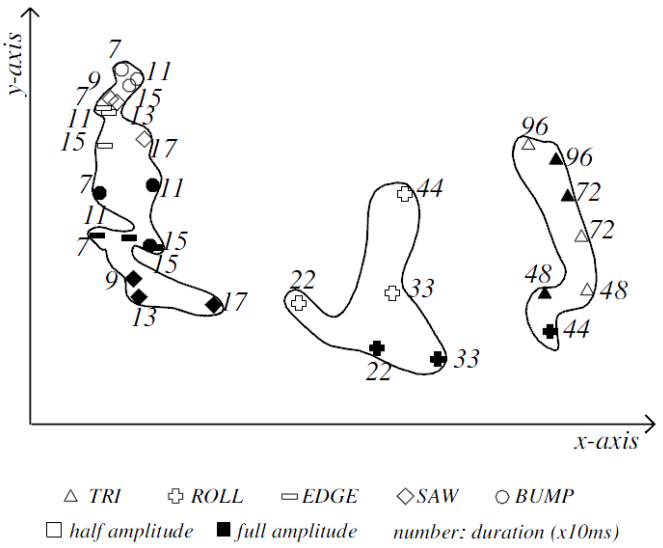


Fig. 7. A Cluster Sorted Multidimensional Scaling Plot Example [44]

A very detailed examinations of the MDS plotting methods usability for haptic icons were done by Pasquero et al.[44] and Luk et al. [40], verifying that MDS plots are robust against noise across the dissimilarity matrix. This states that "it can be a valuable tool to evaluate the expressive capability of haptic devices". Furthermore Pasquero et al. found that "hidden patterns in the resulting plot of a MDS analysis carried out on the entire stimuli set can become apparent when selected submatrices of the full dissimilarity matrix are submitted to the same MDS algorithm. This tends to indicate that the cluster-sorting technique also captures detailed information about sub-level of stimuli distinction that are not visible by sole inspection of the global MDS plot"[44].

Further methods include examination tools like Matlab Simulink [5], step by step supporting shared analysis of common data structures, as demanded by Swindells et al.[54]. Another user study classic the 7-10 level Likert scale might frequently be useful to generate and analyse data, too [42].

3.5 Summary of Technical Expertises

In this section a summary of relevant results for HIP and haptic icon design are brought together.

Designers of haptic feedback should always be aware of the given fact that after a first contact on the skin, the further sensation of superficial structures require relative motion between each other [53]. Enriquez et al.[25] hypothesized about the possibility of 'haptic numbness', which should be taken into account when evaluating performances on haptic interfaces. "In the same manner that 10% of the population is color-blind, there may be a naturally-occurring difficulty in learning haptic stimuli associations"[25].

Apart from that Swerdfeger et al.[53] suggest in their design heuristics for family based design of melodic haptic icons that non-syncopated rhythms should be used primarily, because rhythmic changes dominated other distinctions and where grouped even if amplitude or frequency had been different. Though amplitude and frequency should be used as secondary design parameters. The similar number of perceived notes is an important grouping factor to users, especially groups of rapid eighth notes showed a unique recognition value in rhythm design. But this effect lost distinctiveness even leading to confusion of users, if several haptic icon families used the rapid eighth notes. "As long as they are not emphasized, quarter notes can

be replaced with two eighth notes for within-group variation"[53]. A similar relation like that between rhythm and amplitude/frequency was found by Hoggan and Brewster between waveform (recognition rate of 94.2% [32]) and amplitude/frequency, stating that rhythm and waveform are very important haptic icon design parameters. For mobile devices it turned out to be rhythm and amplitude, which are best suited for distinctive haptic icon design [18]. Furthermore a perceptual segregation of abrupt and 'rolling' rhythms assured a meaningful base for one haptic icon family [53].

Tactile dynamics can be achieved by linear and exponential (de-) crescendos (de-/increasing amplitude) and could be used for haptic icon design as a wise means due to recognition rates of 92-100% [17]. Another musical technique, sforzandos (high intensity accents) thus perpetrated confusion and should not be used in tacton design at the moment.

The training aspects in HIP highly effected the results in many studies. Though training for sforzandos could also sensibilize for such a haptic icon stimuli [17]. Enriquez et al.[25] allowed "participants to return to the self-guided learning interface when consistent mistakes in the enforced learning phase have been detected", with positive influence on the over all results. Another positive effect of training showed the results by Hoggan and Brewster, with crossmodal recognition rates of 85% between Earcons and Tactons. With only three further training sessions participants recognized tactons with rates of 90% or above [31].

A final helpful advice for HIP comes from a comment by Swindells et al.[54]: "Although perceptual sensitivity would be a useful addition to a haptic prototyper, there are many unknowns within the perceptual limitations of haptic behaviors. So, currently, the best approach for haptic prototype design is to perform perceptual user studies to compare several designed haptic behaviors after they have been developed."

4 Future Perspectives

In this section open questions are summed and some inspirational ideas are given in the outlook of unexplored haptic icon feedback technologies.

4.1 Open Questions

Until today scientist have defined the affected parameters of tactons, but still investigate how they interfere with each other. "Parameters which work well alone may not work well when combined with others into a tacton. For example, one parameter may mask another"[13]. So there is a knowledge base about the tactile fundamentals, but the actual composition of haptic icons still needs further research.

The "number and complexity of required haptic syllables" is still questionable say Enriquez et al.[25]. They also assume that avoiding mid-values in hatpic phoneme sets would "sufficiently improve identification performance to justify additional dimensions to increase set size". It then would be easier for people to distinguish haptic icon sets with 3 dimensions with 2 values on each (2^3=8) rather than 2 dimensions with 3 values (3^2=9) [25]. Also, how "temporal information affects absolute identfication of tactile rhythm"[42] needs further scientific evidence. "In addition, future studies could investigate the perception of more complex amplitude changes"(parabolic shapes), as much as, how other musical techniques like the (de-)crescendos can be applied to the field of hatpic icon design [17]. Identification of haptic icons encoded by roughness and containing short notes turned out difficult, due to the short time to identify roughness. It will need future Work to define the minimum duration of notes at which a roughness parameter can still be identified, so "rhythms could then be designed around these results"[15], even though roughness is not a suitable tacton parameter "when using mobile phone vibration motors"[18].

"The concurrent presentation of multiple tactons must also be studied. These studies will answer some of the main questions regarding the usability of tactons and a good understanding of their design and usability will have been achieved"[13]. According to the underlying data structures, Swindells et al.[54] ask for "more explicit shared data structures and handles for integration with other modalities, such as video and audio". This would support the development of tightly pleached multimedia devices a lot.

4.2 Unexploered Haptic Icon Feedback Technologies

As much as the technical parameter 'temperature' *(see 2.5 Technical Parameters of Haptic Icons)* hasn't been used yet, there are some more ways of creating haptic icon effects, no one in the studied works has thought off. For instance, one would be an adjusted massage recliner chair used as sketching hardware for drawing spatial haptic patterns over user bodies.

Another one is only lightly mentioned by Swindells et al.[54], who say that "one could use surfaces of silk, wood, sandpaper, and metal to express how a mechatronic tactile system might feel when in particular system states"[54]. Unfortunately material state or surface metamorphosis hardly exists so far, especially for electronic devices. Chimeras like that would need strong team work of chemists, electronic engineers, computer scientists etc., to find out about the potential benefits. Only something like ferrofluids come close to those characteristics. This is a fluid mixture which can be produced out of motor oil and toner of a laser printer. The amazing characteristic of being liquid and magnetic allows to control the ferrofluid. This originally was invented by the NASA to control fluids and fuels in zero gravity. One could maybe think of a slight film of ferrofluid being in front of a touchscreen. The ferrofluid then is converged by magnetic fields to give a haptic feedback directly at the position on the surface of the touchscreen, where it is needed. This should just give some inspiration and encouragement for future technologies, maybe being useful to represent haptic icon effects.

5 Conclusion

In the young time tactons have been in the spotlight of research for haptic feedback and user interfaces, they could already present their positive impacts on a big range of HCIs. Several hi-fi and lo-fi prototyping methods showed that they are useful techniques in prototyping haptic icons. The proven concepts of icons in visual and auditory HCI is a central theme through all literature and papers of haptic icons, telling us once more that the reusability of successful realized techniques help a lot to explore new fields. With the expected improvement in hapitc icon prototyping, also the designers in electronic companies will be strengthened to justify the effort in creating new ways of transmitting informations to the users. The advantages of balancing informations on multiple cognitive human channels, will have two aspects. First of all it hopefully results in greater satisfaction of handling future technological devices, providing for example that also disabled people have more ways to communicate. On the other side complex coherencies, like creating certain feelings for entertainment purposes, maybe only get perceivable by the simultaneous perception through all sensory channels at once.

A main question in HIP definitely is, if it is more suitable to prototype haptic stimuli and their contextual device in which they are displayed separately. At which point of testing should designers prototype them together? Since the beginning? These questions have a great impact on budget and time schedules, depending really on the specific prototyping case. Some punchy tactons might not have the same expressiveness when played under likely conditions. Therefore testing hatpic icons should primarily be done in the most realistic circumstances in my opinion.

In the end Cottam and Wray phrased the best final appreciation for the current state of HIP research aspects: "Tangible interfaces can be more natural, intuitive, and efficient than the way we currently interact with digital devices and interfaces. The key to developing these innovations is the ability to use physical-computing materials early during sketching. To innovate new methods of interaction between analog and digital interfaces, we must be able to not only imagine them but also explore, observe, and demonstrate them. Besides the advantages of observing an interaction in a physically real context, there's undeniable benefit in the ability to show tangible-interface concepts and technologies to others, rather than just describing them"[22]. The summary

in this work tries to contribute its part in the development of suitable haptic icon prototyping techniques and guidelines.

REFERENCES

[1] http://www.naturstudiendesign.de/bilder/Der_Mensch/70_Haut/37_Tastsinn.JPG. visited 03.01.2010.

[2] http://en.wikipedia.org/wiki/Optacon. visited 03.01.2010.

[3] www.tiresias.org. visited 03.01.2010.

[4] Manual Nokia Digitalpen (SU-27W). http://nds1.nokia.com/phones/files/main_page/Nokia_SU-27W_UG_de.pdf. visited 03.01.2010.

[5] The mathworks matlab and simulink. http://www.mathworks.com. visited 11.01.2010.

[6] A. Chan, K. MacLean, and J. McGrenere. Learning and identifying haptic icons under workload. WHC, 2005. pages 432-439.

[7] Audiological Engineering Corporation. http://www.tactaid.com/. visited 03.01.2010.

[8] O. Bau, U. Petrevski, and W. Mackay. Bubblewrap: a textile-based electromagnetic haptic display. In *CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 3607–3612, New York, NY, USA, 2009. ACM. good documentary QuickTime Video available, visited 20.11.2009.

[9] S. Biggs and M. Srinivasan. Haptic interfaces. *Handbook of Virtual Environments: Design, Implementation, and Applications*, K.M. Stanney, ed.,Lawrence Erlbaum:93–115, 2002.

[10] H. V. Bjelland and K. Tangeland. User-centered design proposals for prototyping haptic user interfaces. http://www.springerlink.com/content/4vu1127kg1g48569/fulltext.pdf?page=1 http://www.springerlink.com/content/4vu1127kg1g48569/fulltext.pdf?page=2 http://www.springerlink.com/content/4vu1127kg1g48569/fulltext.pdf?page=3, 2007. 3 pages as preview available, visited 20.11.2009.

[11] M. M. Blattner, D. A. Sumikawa, and R. M. Greenberg. Earcons and icons: Their structure and common design principles (abstract only). *SIGCHI Bull.*, 21(1):123–124, 1989.

[12] S. Brewster. Flickr set of the nordichi 08 workshop on lo-fi haptic prototyping. http://www.flickr.com/photos/24420490@N08/sets/72157608986634676/. visited 21.12.2009.

[13] S. Brewster and L. M. Brown. Tactons: structured tactile messages for non-visual information display. In *AUIC '04: Proceedings of the fifth conference on Australasian user interface*, pages 15–23, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[14] S. A. Brewster and L. M. Brown. Non-visual information display using tactons. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 787–788, New York, NY, USA, 2004. ACM.

[15] L. M. Brown, S. A. Brewster, and H. C. Purchase. A first investigation into the effectiveness of tactons. In *WHC '05: Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 167–176, Washington, DC, USA, 2005. IEEE Computer Society.

[16] L. M. Brown, S. A. Brewster, and H. C. Purchase. Multidimensional tactons for non-visual information presentation in mobile devices. In *MobileHCI '06: Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 231–238, New York, NY, USA, 2006. ACM.

[17] L. M. Brown, S. A. Brewster, and H. C. Purchase. Tactile crescendos and sforzandos: applying musical techniques to tactile icon design. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 610–615, New York, NY, USA, 2006. ACM.

[18] L. M. Brown and T. Kaaresoja. Feel who's talking: using tactons for mobile phone alerts. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 604–609, New York, NY, USA, 2006. ACM.

[19] C. Magnusson, S. Brewster. Nordichi 2008 workshop: Guidelines for haptic lo-fi prototyping. http://www.english.certec.lth.se/haptics/lo_fi_nordichi_2008.htm, October 2008. visited 21.12.2009.

[20] A. Chang and C. O'Sullivan. Audio-haptic feedback in mobile phones. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1264–1267, New York, NY, USA, 2005. ACM.

[21] R. Cholewiak and M. Wollowitz. The design of vibrotactile transducers. *Tactile Aids for the Hearing Impaired, Whurr Publishers Ltd: London*, I. Summers, ed:57–82, 1992.

[22] M. Cottam and K. Wray. Sketching tangible interfaces: Creating an electronic palette for the design community. *IEEE Comput. Graph. Appl.*, 29(3):90–95, 2009.

[23] Engineering Acoustics Inc. Tactor products. http://www.eaiinfo.com/Tactor%20Products.htm. visited 03.01.2010.

[24] M. Enriquez. Perceptual design of haptic icons. EuroHaptics, 2003.

[25] M. Enriquez, K. MacLean, and C. Chita. Haptic phonemes: basic building blocks of haptic communication. In *ICMI '06: Proceedings of the 8th international conference on Multimodal interfaces*, pages 302–309, New York, NY, USA, 2006. ACM. visited 20.11.2009.

[26] M. J. Enriquez and K. E. MacLean. The hapticon editor: A tool in support of haptic communication research. In *HAPTICS '03: Proceedings of the 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'03)*, page 356, Washington, DC, USA, 2003. IEEE Computer Society.

[27] J. Gill. Guidelines: Pictograms, icons and symbols. Royal National Institute of the Blind, UK, 2003. Available at http://www.tiresias.org/research/guidelines/index.htm, visited 10.01.2010.

[28] D. G. Gunther, E. and S. O'Modhrain. Cutaneous grooves: Composing for the sense of touch. In *Proceedings of Conference on New Instruments for Musical Expression, Dublin, IR, 1-6*, 2002.

[29] E. S. Gunther. A tool for composition in the tactile modality. Master's thesis, Department of Electrical Engineering, MIT, Boston, MA, 2001.

[30] K. S. Hale and K. M. Stanney. Deriving haptic design guidelines from human physiological, psychophysical, and neurological foundations. *IEEE Comput. Graph. Appl.*, 24(2):33–39, 2004.

[31] E. Hoggan and S. Brewster. Designing audio and tactile crossmodal icons for mobile devices. In *ICMI '07: Proceedings of the 9th international conference on Multimodal interfaces*, pages 162–169, New York, NY, USA, 2007. ACM.

[32] E. Hoggan and S. Brewster. New parameters for tacton design. In *CHI '07: CHI '07 extended abstracts on Human factors in computing systems*, pages 2417–2422, New York, NY, USA, 2007. ACM.

[33] E. Hoggan, A. Crossan, S. A. Brewster, and T. Kaaresoja. Audio or tactile feedback: which modality when? In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 2253–2256, New York, NY, USA, 2009. ACM.

[34] E. E. Hoggan and S. A. Brewster. Crossmodal icons for information display. In *CHI '06: CHI '06 extended abstracts on Human factors in computing systems*, pages 857–862, New York, NY, USA, 2006. ACM.

[35] H.Z. Tan, N. I. Durlach, W.M. Rabinowitz, C.M. Reed. Information transmission with multi-finger tactual display. Research Laboratory of Electronics, Masachusetts Institute of Technology, Cambridge, MA, USA, 1997.

[36] I. R. Summers, C. M. Chanter, A. L. Southall and A. C. Brady. Results from a tactile array on the fingertip. In *Proceedings of Eurohaptics 2001, Birmingham*, pages 26–28. University of Birmingham, UK, 2001.

[37] Immersion VibeTonz System. http://www.immersion.com/markets/mobile/index.html. visited 09.01.2010.

[38] M. Kretz. Article about thermoreception. http://www.studentenlabor.de/ss04block/thermorezeption.htm, 2004. visited 13.01.2010.

[39] D. J. Lewkowicz. The development of intersensory temporal perception: An epigenetic systems/limitations view. *Psychological Bulletin*, 126:130–155, 2000.

[40] J. Luk, J. Pasquero, S. Little, K. MacLean, V. Levesque, and V. Hayward. A role for haptics in mobile interaction: initial design using a handheld tactile display prototype. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 171–180, New York, NY, USA, 2006. ACM.

[41] M. Hollins, R. Faldowski, S. Rao, F. Young. Perceptual dimensions of tactile surface textures: A multidimensional scaling analysis. *Perception & Psychophysics*, 54 (6):697–705, 1993.

[42] T. L. McDaniel, S. Krishna, D. Colbry, and S. Panchanathan. Using tactile rhythm to convey interpersonal distances to individuals who are blind. In *CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 4669–4674, New York, NY, USA, 2009. ACM.

[43] D. K. McGookin and S. A. Brewster. Understanding concurrent earcons: Applying auditory scene analysis principles to concurrent earcon recog-

nition. *ACM Trans. Appl. Percept.*, 1(2):130–155, 2004.

[44] J. Pasquero, J. Luk, S. Little, and K. MacLean. Perceptual analysis of haptic icons: an investigation into the validity of cluster sorted mds. *Haptic Interfaces for Virtual Environment and Teleoperator Systems, International Symposium on*, 0:67, 2006.

[45] R. W. Cholewiak, J. C. Brill, A. Schwab. Vibrotactile localization on the abdomen: effects of place and space. *Perception and Psychophysics*, vol. 66:970–987, 2004.

[46] J. Rovan and V. Hayward. Typology of tactile sounds and their synthesis in gesture-driven computer music performance. *Trends in Gestural Control of Music*, pages 297 – 320, 2000.

[47] A. Rupert. Tactile situation awareness system: proprioceptive prostheses for sensory deficiencies. *Aviation, Space and Environmental Medicine*, 71:92–99, 2000.

[48] S. E. Newman, A. D. Hall, D. J. Foster, and V. Gupta. Learning as a function of haptic discriminability among items. *The American Journal of Psychology*, 97(3):359372, 1984.

[49] C. Sherrick. A scale for rate of tactual vibration. *Journal of the Acoustical Society of America*, 78, 1985.

[50] C. Sherrick and R. Cholewiak. Cutaneous sensitivity. *Handbook of Perception and Human Performance: Sensory Processes and Perception*, v.1, 1986. K. Boff, L. Kaufman, and J. Thomas, eds., John Wiley & Sons.

[51] C. M. Smith. Human factors in haptic interfaces. *Crossroads*, 3(3):14–16, 1997.

[52] I. Summers. Single channel information transfer through the skin: Limitations and possibilities. In *Proceedings of ISAC 2000*. University of Exeter, UK, 2000.

[53] B. A. Swerdfeger, J. Fernquist, T. W. Hazelton, and K. E. MacLean. Exploring melodic variance in rhythmic haptic stimulus design. In *GI '09: Proceedings of Graphics Interface 2009*, pages 133–140, Toronto, Ont., Canada, Canada, 2009. Canadian Information Processing Society.

[54] Swindells, Colin and Maksakov, Evgeny and MacLean, Karon E. The role of prototyping tools for haptic behavior design. In *VR '06: Proceedings of the IEEE conference on Virtual Reality*, page 90, Washington, DC, USA, 2006. IEEE Computer Society.

[55] T. McDaniel, S. Krishna, V. Balasubramanian, D. Colbry and S. Panchanathan. Using a haptic belt to convey non-verbal communication cues during social interactions to individuals who are blind. *HAVE*, pages 13–18, 2008.

[56] J. van Erp and M. Spapé. Distilling the underlying dimensions of tactile melodies. In *Proceedings of Eurohaptics 2003*, pages 111–120, 2003.

[57] H. v. Veen and J. van Erp. Tactile information presentation in the cockpit. In *Proceedings of the First International Workshop on Haptic Human-Computer Interaction*, pages 174–181, London, UK, 2001. Springer-Verlag.

[58] R. T. Verrillo and G. A. Gescheider. Perception via the sense of touch. *Tactile Aids for the Hearing Impaired*, I.R. Summers (Ed.):1–32, 1992.

[59] S. B. William W. Gaver, Jacob Beaver. Ambiguity as a resource for design. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, April 2003. Ft. Lauderdale, Florida, USA.

Prototyping of Interactive Surfaces

Martin Hommer

Abstract— Interactive surfaces like tabletops, wall-mounted screens or mobile touch displays, are not just an affair of the research anymore but now appear numerously in our daily life and thus are being used by more and more people. This entails that research must not only concentrate on exploring new sophisticated features but also may well design systems with less functionality and in return as much more usability. This is the motivation for this paper or, more precisely: how can prototyping help designers of an interactive surface to work more user-oriented and furthermore does it help them to make the design process more efficient? This work will introduce the topic from a general point of view, go into details of prototyping tools which are applicable for this topic and describe how prototyping has been used in recent research projects, which are dealing with interactive surfaces or corresponding interaction techniques. Thus, this paper provides an overview of related research work as well as a discussion about their proceedings, in order to analyze what they did well and where potential improvements can be made.

Index Terms—Prototyping, Interactive Surface, Display, Interactions, User-Centered Design, Iterative Design, User Study, Evaluation

1 Introduction

As they became smaller and their technology smarter, displays got out of standard computer monitor cases and are spreading into our daily environment and now support the vision of ubiquitous computing. They are found at office doors, in coffee tables, in cars and among a lot other places and objects, also in mobile phones. Frankly, a phone is equipped with a small screen not only since yesterday, but with devices such as the *iPhone* [1], a new way of interaction has become omnipresent: direct touch. This input method allows a more natural way of utilization than using a mouse or a touchpad. However, this is just the tip of the iceberg and new means of man-machine-communication have been researched [31, 32, 8] *(see figure 1b for an example)* and a lot more are about to be.

Thus, the requirements on these displays and the corresponding interaction techniques as well as the freedom of design are huge, but so are the mistakes a designer can make. A technically sophisticated design is futile, if a user has difficulties to handle it, because its way of usage is unnatural. So, the design process of an interactive system should integrate the user to get early feedback. This is one of the goals of prototyping.

1.1 What is Prototyping?

This work deals with the examination of prototyping techniques for interactive surfaces, but before, the terms *prototyping* and *interactive surface* should be regarded in principle. When hearing the word *prototype*, one often thinks of a product being nearly finished and ready to be reproduced several times. But actually this is only half the truth as a prototype can as well be a model in an earlier design phase [5] which solely provides a subset of the subsequent functions. Thus, from this definition the term *prototyping* can be reasoned as the process of designing and evaluating prototypes throughout several design phases and integrate the particular results in new prototypes.

Prototyping is widespread in the field of HCI and helps designers of both software and hardware to evaluate their products during the development process. However, a common mistake is to believe that everything prototyping is about is testing a system's usability, for example in a user study, in order to reveal failings [21]. Though this is one of its most important objectives, far more time is spent with prototyping to build a basis for designers to "organically and evolutionarily learn, discover, generate, and refine designs" [21]. Thus, a constructive reconsideration of a design can be stimulated by prototyping and so, finding new features can be controlled instead of left to chance. Because prototypes are apprehensible and concrete, whether they are a real tangible object or a piece of software, it is easier to discuss on them within a project team and it is more likely that new solutions or ideas for novel features arise (like in [9, 19]). Moreover, comparing and choosing between different designs is easier when you have presentable alternatives than merely a rough vision in your mind.

However, prototyping in association with user experiences is also a very important part of the design procedure, because integrating user feedback in early stages of a process reveals usability and acceptability issues, which then can be corrected in good time. In addition, developers can estimate whether vital or sophisticated features are persuasive and aside from that, the user might be encouraged to think about new ones. Benefits from an economic point of view are also existent, as for one thing prototyping reduces development costs, for example when using rapid prototyping *(see figure 1a)*. For another thing, prototypes can be used to better convince a possible investor or a principal - for the same reason as arguing within the project team is improved: tangibility and demonstration.

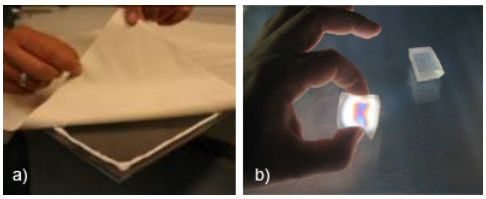


Fig. 1. *a)* Rapid prototyping of a multi-touch interactive surface [14]. *b)* New ways of interaction are being explored [31].

1.2 What are Interactive Surfaces?

The term *interactive surface* includes not only displays in the narrower sense, but also non-digital surfaces, like a wall or a desk, where images are projected on. They will be described as displays as well.

As mentioned before, they appear in various shapes. The most common ones are tabletops, where the screen is mounted horizontally and which can usually be reached from all sides. Normally, their height is comparable to the one of a coffee table (like the *Microsoft Surface* [2]) or a desk (like *EnhancedDesk* [19]), depending on the desired use. Next to tabletops, there are also vertically mounted displays. They have basically different characteristics which are to be heeded. For example, because horizontal surfaces have no upper side, the orientation of the content depends on each user's position and thus is often upside down for other viewers. A wall mounted screen does not need to deal with it. In contrast, hanging displays sometimes suffers from a bad attainability, if they are too large, so that their total height can't

- *Martin Hommer is studying Media Informatics at the University of Munich, Germany, E-mail: martin.hommer@gmx.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009/10*

be reached, or too small and thus only a few people can physically access them at a time. Also the ergonomic challenges diverge: While a person interacting with a wall mounted display is likely to suffer from physical fatigue in his arm, a tabletop might cause its user neck strain when looking down on it. In a field study [26], which explored the use of touch displays in offices (both horizontally and vertically) for one month, seven of eight participants preferred the vertical ones. They claimed that leaning over the level screen was uncomfortable, especially when reading text. This result seems comprehensible because in this case, the vertical screens were standing on the desk like normal flat screen monitors so that elbows could be rested on the table, which thus prevents arm fatigue. Though this is only a small exemplary comparison, it shows that a lot of factors have to be taken into account when designing an interactive system, to avoid significant limitations of use.

As mentioned above, direct touch is a widespread way of communicating with an interactive system. Due to the shared device for in- and output, the interaction is more natural. A direct-touch interface can also be a *multi-touch* and *multi-user* interface. Multi-touch means that more than one input can be recognized at a time - for example pressing two buttons simultaneously. Beyond that, a multi-user system is able to assign each input to the respective user. But aside from direct touch, there are far more ways of input which have been explored in recent research and novel ones are about to follow. A few existing examples for interaction technologies are Tangibles [28, 35, 31], integrating paper and touch on a tabletop [19] or embodied interaction, where the location and orientation of a human body, as well as its head orientation is used [36].

Thus, a research targeting the prototyping of interactive surfaces should not stop at their design but ought to deal with the respective interaction technique as well. Hence, this paper targets both subjects. What this work is particularly not aiming for is the design of graphical user interfaces for interactive displays, since this is a distinctly different research topic, where clearly differing prototyping techniques come into play. After this introduction, prototyping will be examined in more detail, by presenting common prototyping techniques and key characteristics. Subsequently, some existing prototyping tools are introduced, which can possibly be integrated in a particular design process. Afterwards, the usage of prototyping in practice will be examined and thus several recent research projects are presented and investigated to figure out whether, how, why or why not prototyping has been used, as well as possible accompanying benefits or problems. In the end, the main findings will be discussed, followed by the conclusion.

2 Prototyping in Detail

Various types of prototyping techniques exist, however some are very specific. This section provides an introduction to the different characteristics of prototypes as well as common prototyping methods. Finally, a classification of these methods is given, regarding their suitability for interactive surfaces.

2.1 Characteristics of Prototypes

There are many kinds of prototypes which can appear in a design process. To better differentiate them from another, some general characteristics can be determined.

Fidelity. The *fidelity* predicates the degree of similarity of a prototype and the finished product. Thus a low-fidelity prototype has less in common with the final version than a high-fidelity exemplar. To take an example, some rough sketches of an interaction possibility would have a low fidelity whereas an implemented version of that interface which is able to be used, has a higher one. Certainly both have advantages: prototypes with less details are much quicker to build and thus are cheap. Therefore they are more applicable for early stages of the design process since more different ideas are created and evaluated fast. High-fidelity prototypes are rich in detail and thus have more in common with the later product. This leads to more qualitative results but adaptions are harder to realize. If possible, an optimal design process should integrate a couple of prototypes, starting with a lower fidelity which is increased incrementally.

Level of detail. Another characteristic to be mentioned, is whether the technique is concentrated in a *vertical* or *horizontal* level of detail. This means that a horizontal prototype has the same amount of features as the future product (quantitative conformity) but each of these is not fully implemented and thus does not provide the full functionality. The other alternative implies just a few of the desired functions but these are completely attached (qualitative conformity). Again, both options have benefits: If you want to present all the desired capabilities, though only in outlines, horizontal prototyping is appropriate. The other alternative is chosen if only a subset of features should be explored - these however with the later functionality.

Utilization in the design process. Prototypes have different uses in the whole development process: First, there are *throw-away prototypes* which - as the name implies - will be discarded after their evaluation. Next are *incremental prototypes*, being used when the final product is divided into smaller parts which are developed separately and at last, when employing *evolutionary prototypes*, each one serves as a basis for the next one, whereas every single one is not thrown away [10].

2.2 Methods of Prototyping

There are several methods of prototyping existing, the most common ones should probably be *paper prototyping*, *mock-up prototyping*, *wizard of Oz prototyping* and *video prototyping* [20].

Paper prototyping. A *paper prototype* visualizes the look and/or the content of the possible system by means of paper. It is often utilized when designing user interfaces, because some example pages of the system can be made easily and by cutting the sketches in parts, the interface can be rearranged very quick.

Mock-up prototyping. The next practice is *mock-up prototyping* where, as well as with paper prototyping, a low-detail copy is used, which can consist for example of polystyrene. An example for the use of mock-up prototyping is when designing a mobile phone, where its size should be evaluated and thus no working functionalities are needed. If such small to medium sized hardware should be prototyped, also the use of a 3D printer is appropriate due to its possibility of rapidly building dummies which are looking very much like the finished product.

Wizard of Oz prototyping. When some of a system's core functionalities are not implemented but taken over by a person, without the test subject knowing about it, it is called *Wizard of Oz prototyping*. A common example scenario is an event handler of a user interface which normally reacts on a user's input and updates the interface's content, but in this case, this job is done by an assistant who manually does the updating. This process can be deployed whenever some switches and levers can be shifted by a human to avoid a complex and time-consuming implementation in earlier stages.

Video prototyping. When doing *video prototyping*, an example usage of a system is filmed. Test subjects can not interact with this prototype but they can give their opinions as they can see the designated use of the system. This kind of prototyping can be combined with mock-ups, as the objects used in the video can absolutely be fake and unfinished[1].

For the design and development of interactive surfaces, some of these techniques are particularly suitable. For testing proper finger recognition of a system, one should use a prototype with a higher fidelity which probably does not support the whole set of features but one or two, however these in high details (vertical prototyping). Moreover, that prototype should be in a well-engineered form to enable qualitative exploration. In earlier stages, when parameters like the size or the orientation of a display are to be determined, low-fidelity prototypes like paper or mock-ups are advisable, because their use is cheap and they can quickly be adapted. Methods like Wizard of Oz or video prototyping are rather targeting areas concerning user interfaces. They are less applicable for the interactive surface itself but as much more for prototyping interaction techniques.

[1] More information about video prototyping and an example usage can be found in Tognazzini's "starfire" project [34].

3 Prototyping Tools

This section introduces some examples of existing prototyping tools, which are mainly targeting the design of interactive surfaces or pure interaction techniques. Since there is basically no allround prototyping tool for these topics, different ones will be presented from the particular related fields.

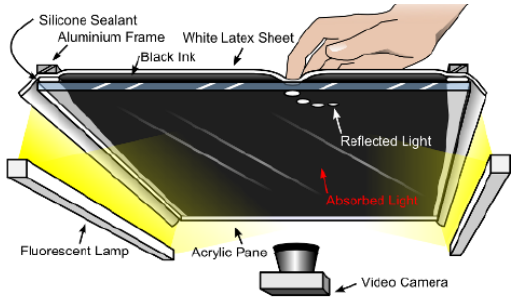


Fig. 2. Technical concept of the Malleable Interactive Surface [14]. When pressing the white latex, black ink is pushed away so that a camera beneath recognizes the appearing white spot as the touch of a finger.

3.1 Malleable Interactive Surfaces

Hilliges et al. [14] have introduced an approach for rapidly prototyping surfaces which are capable of sensing both multi-touch and objects. In their paper, they describe a simple but efficient technique to achieve this, which copes without complicated electronics but instead uses off-the-shelf hardware. It is based on a white latex sheet, a (plexi)glass plate and black ink poured in the space between these two surfaces. Input works via pressing the malleable latex which then pushes away the liquid beneath and reveals its white color to the camera below the glass plane *(see figure 2)*. Thus, a touch itself as well as its strength is recognized by the brightness of one or several spots in the captured image. The content the user sees, is projected on the textile from above (front-projection). This technique allows a quick building of an interactive surface prototype (due to the gravity limited to a horizontal use). It can be utilized for various purposes and stages in the design process, like prototyping gestures, exploring different dimensions for a tabletop and their extent for usage or as a testing environment for interactive applications.

3.2 OIDE

The *OIDE* (Open Interface Development Environment) is part of the *OI* (OpenInterface) and offers a platform for prototyping multi-modal interaction [25]. The *OI* project's website [3] provides some example videos about the system's various uses. *OIDE* can be used by designers, who want to test and examine different interaction methods during the design process. This system provides a graphical drag-and-drop interface where different devices and actions can quickly be arranged and thus an individual interaction technique can be created. For example, an accelerometer of a device is connected to a slideshow and thus controlling the presentation is done via moving the remote gadget. There are many common devices supported, like Apple's *iPhone* [1], the *Wiimote* (the controller of Nintendo's *Wii* [4]) or mobile phones. The OpenInterface Framework is an open source platform supporting an iterative user-centered design process. It comprises a *kernel*, which links pre-coded components to an application, a *repository*, consisting of component descriptions, interaction techniques and application configurations, a *Forge*, which hosts the software, and the *OIDE*, the graphical tool to link components and applications. Hence, *OIDE* is a tool to build early prototypes, evaluate them and alter quickly their design options or functionalities.

3.3 DisplayObjects

DisplayObjects is a method to enrich physical gadgets with displaying capabilities in order to prototype appropriate devices [6]. This system tracks three-dimensional objects, maps an image, which the object's faked display should show, on the virtual replica and then projects this texture back onto the original surface. Thus, a non-digital mock-up can easily be switched into a display to support rapid prototyping.

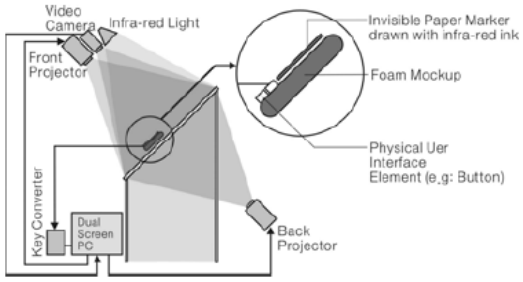


Fig. 3. *ARdesk* [27]. By detecting markers with a camera as well as processing data from physical interface components, mockups are enriched with functionality. Thus both design and use can be evaluated.

3.4 ARdesk

Related but more extensive is the approach of *ARdesk* [27], where similar to the last mentioned project, physical mock-ups are extended with a fake display by using front-projection. Here, the tracking is done via Augmented Reality markers which are stuck onto the objects. Moreover, with the aid of a set of physical user interface components (like for example buttons) and a software toolkit, which can be connected to these input elements and reacts on their events, the product dummies can be enhanced with functionality. On an *Augmented Reality Desk*, the mock-ups are being augmented by a camera and a projector from above and also the desk itself is interactive, due to a projection from behind and the camera above *(see figure 3 for an overview)*. This kind of prototyping tool is mostly applicable for the evaluation of the design and dimensions of interactive objects and only rudimentarily of its functionality.

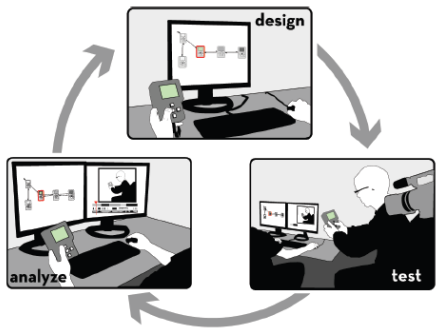


Fig. 4. *d.tools* [13]. The basic design process possible with *d.tools*. Starting with the design of a low- or high-fidelity prototype, developers are then able to run a video-recorded test, which can be analyzed afterwards. Based on findings, the prototype can be redesigned and the cycle starts again.

3.5 d.tools

More regards for the functionality of a prototype has *d.tools* [13] which is a design toolkit software as well as a hardware interface, which links physical components to the software. In the device designer, a device can be built virtually out of controllers (like buttons

or sliders), sensors (like accelerometers or compasses), displays and speakers. The statechart editor extends the virtual prototype with functionality by giving the opportunity to add states and content to the particular components. To expand this to a high-fidelity prototype, physical counterparts of the elements are attached to a mock-up and connected to the hardware interface. Thus, the tangible prefiguration can be executed and evaluated, controlled by the assembled application on the computer. Moreover, *d.tools* provides an analysis tool, where recorded video clips from one or more testing phases can be reviewed, organized and evaluated. Therefore, this system supports an interactive and design-centered development process *(see figure 4)* and is suitable for high- and low-fidelity prototyping.

3.6 $ 1 Recognizer

One section which is often part of the development of interactive surfaces is the recognition of gestures. To rapidly integrate gestures into an application in order to test interactive systems, the *$ 1 Recognizer* [40] can be used. It consists of about 100 lines of code and processes captured images in order to recognize gesture patterns. In their current version, a set of 16 different patterns can be identified. Thus, a designer of an interactive system can quickly integrate interaction through gestures without spending much time for implementing the recognizing technology on his own. In later process stages, either a whole new detection system can be implemented or the *$ 1 Recognizer* can be adapted to the specific needs. The usage of existing components when developing a system can often be valuable, though the regarded piece can hardly be adopted to the particular needs. But when major design or interaction issues are not settled yet, implementing further parts which are based on uncertain solution can be a waste of time if these decisions are overturned later.

4 Prototyping in Practice

This section provides a closer look to the application of prototyping in recent research projects or in other words: how did their design process look like? This paragraph is divided in the consideration of the *design* of interactive surfaces on the whole as well as of projects which primarily dealt with the exploration of *interactions* with appropriate surfaces.

4.1 Prototyping Interactive Surfaces

A lot of work in the research dealt with interactive surfaces. However, not many really did employ prototyping techniques in their design process. The following introduces those projects, where prototyping has been used and presents how they utilized it as well as the resulting consequences.

4.1.1 DiamondTouch

In 2003, Dietz and Leigh from Mitsubishi Electronic Research Laboratories (MERL) have presented *DiamondTouch* [9], a capacitive touch technology, which is not only able to detect multi-touch but also multi-user. This is achieved through receivers, one connected to each user's chair, to assign a contact with the surface to the appropriate user.

In the context of a research on a different project at the MERL, some collaborative meetings have been held with a ceiling-mounted projector displaying content on a table and a single wireless mouse being passed around for interaction. Due to the limitations and the low usability the researchers have noticed while working therewith, they decided to build an interactive surface, which should support simultaneous and also *direct* input. At first, they drew up requirements their system should meet and afterwards searched for the suitable hardware. Based on these findings, they built a prototype out of off-the-shelf components, which has not nearly the size of a collaborative table (20 x 20 cm), but serves as a basis to evaluate the implemented hard- and software. With the aid of this high-fidelity prototype, they were able to test whether their demands were assembled well or if there were shortcomings *(see figure 5)*. During the evaluation, they found an issue regarding false detection of a touch signal, caused by varying noise levels and interferences when chairs are dragged. Moreover, they learned that objects being placed on the surface do not affect the sensing, even if they are of metal. Thus, they came across to ponder whether to build objects in the future which do influence the table.

This project is an example for the use of prototyping for an internal purpose and not for the integration of the user. However - and due to the fact that a lot of projects have used the *DiamondTouch* technology - it may be assumed that they considered users' opinions in later process stages.

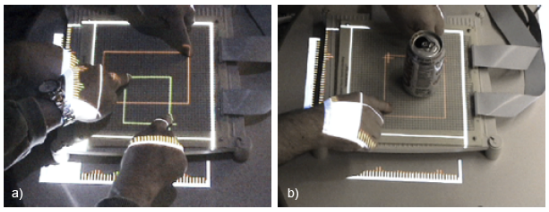


Fig. 5. *DiamondTouch* [9]. Photographs of experiments with the first high-fidelity prototype of the *DiamondTouch* technology, showing *a)* the multi-user support and *b)* the resistance against conductive objects.

4.1.2 EnhancedDesk

Next to capacitive sensing, as it is employed in *DiamondTouch*, another common method to recognize user input is via image processing, where a computer tries to detect fingertips or palms from a captured video frame. This technique was utilized in the *EnhancedDesk* project [17, 19], where besides a human hand also paper sheets equipped with markers are identified. The goal was to build an office desk on which analogue and digital media is smoothly integrated and where digital media can be directly manipulated with the finger or hand.

The first prototype was built in 1998 [17], referring to Wellner's *DigitalDesk* (1993) [38]. This was similar in its hardware configuration, consisting of a CCD[2] camera above a white desk and a video projector. By experimenting with this prototype, Koike et al. have noticed three main issues [19]: for one thing, the detection of a hand often fails, because the image projection from above implicates a change in color of both the skin and the background so that color differentiation is unreliable. For another thing, finger recognition is not done in real-time and causes lags. However, in order to replace mouse input by fingers, the response time should be drastically lowered. Additionally, the marker on the paper, which the system uses for detection, has to be very large due to the low resolution of the video camera.

Based on these findings, a second prototype has been developed in 2001, differing in the technology of the camera and the image processing software. For the detection of hand and fingers, an infrared camera is used now, which provides thermal images and thus the recognition is not disturbed by overlay projections. The necessary size of a marker's surface has been reduced by 84% due to a pan-tilt camera, which can zoom in to the paper. Also the input responses are nearly done in real-time now.

Thus, the *EnhancedDesk* is another example for the internal usage of prototyping to help designers discover problems during the design process, so that they can be eradicated in the subsequent implementation. This is emphasized by an anecdote: "In our first prototype we also experienced that the system worked properly in the morning, but did not in the evening." [19]

4.1.3 Hermes, SPAM

As mentioned earlier, an important use of prototyping is to integrate the user in the design process of interactive systems. This has been done multiply in the development procedure of *Hermes 1*, *Hermes 2* and *SPAM* [11]. During their work, the involved researchers found out that not only technical feasibility should be evaluated but "it's often equally important to investigate factors such as use and appropriation and that in some cases, without user studies, technical feasibility can be meaningless" [11].

[2] Charge-Coupled Device.

Hermes is a set of interactive office door displays where digital messages can be left or sent to another screen. For their first approach, *Hermes 1*, they split their development process in several phases, each one applying iterative prototyping. They decided to implement only a small set of features (vertical prototyping) and to run the test over a longer period (with each phase lasting about four months). The first phase targeted core functionalities and has been evaluated by the project's staff itself, in a real scenario outside their laboratory. This trial revealed the significant error of humans blocking wireless network signals, a typical case of an issue not being apparent before a test run. Their next two phases implied an increase in the amount of screens as well as a stronger reliability and more means of interaction. Moreover, displays were given to external people to get more qualitative user feedback. Their overall testing period lasted nearly three years, with continuous further improvements. Remarkably is that they split up each single phase into prototyping cycles which allowed them to redesign and evaluate their prototypes on and on.

Subsequently, *SPAM* (SMS Public Asynchronous Messenger) was developed, supporting messaging between displays and/or mobile phones over a longer distance, based on the common short message service. In an initial design workshop, they encouraged possible users to discuss requirements and solutions, by giving them pre-defined scenarios and props (like the *Hermes 1* display). Based on the workshop's results, they quickly built a prototype using off-the-shelf hard- and software, followed by testing. Finally, they deployed it in two locations for user evaluations.

After developing *Hermes 1* and *SPAM* and gaining a lot of knowledge about user needs, the researchers aimed to design a second version of *Hermes* up from the bottom. Thus, they wanted to define basic characteristics completely new, namely the physical form factor and the display configuration. Again, they used off-the-shelf hard- and software to rapidly prototype several alternatives.

Using standard products for rapid prototyping has great benefits like quick and cheap development and proven reliability. However, as these technologies are often targeting a different use, they have to be tailored. Thus, enormous investigations and testing have to be made resulting in a drastically decrease in rapidity.

4.1.4 ColorTable

The *ColorTable* [22] is another good example of a well thought-out design process, consisting of workshops proceeding in an iterative design-evaluation-feedback-redesign procedure. This project aims at supporting urban planners by means of TUIs[3] and a mixed-reality environment. Users should be able to select and position objects on a real map and see a video of that place with the desired object rendered into it.

In total, three prototypes have been built, each one being high-fidelity and fully usable. After each evaluation, they were able to spot the shortcomings of their current prototype as well as features which were be nice to have. For example, when testing the first version, people stated that they are desirous of moving the viewport. This led to a redesign of the surface and an implementation of a rotating table.

The researchers focused on deploying prototypes with a higher fidelity and thus can profit from more qualitative evaluation results. But in contrast, this results in a longer time of production and especially in early stages, where basic ideas are likely to be discarded (as the introduction of the rotating table shows), this is ineffective.

4.1.5 Curve, BendDesk

When designing interactive surfaces, also ergonomic aspects should not be disregarded. Two current projects are dealing with developing an ergonomic tabletop including both a vertical and a horizontal screen which are combined to one curved display. It is argued that both orientations have advantages and thus a combination of them would make it possible to benefit from both [39]. Long-term studies about the use of horizontal and vertical displays confirm the ergonomic impact of the screen's orientation. From a survey of 58 tabletop researchers emerged that purely horizontal use causes back and neck strain [15] and also a field study revealed similar results, where normal offices were enriched with horizontal or vertical displays to explore their benefits and problems [26]. One person even stated that it would be ideal if "you could switch between horizontal and vertical positions depending what you're doing" [26].

The *Curve* project [39] *(see figure 6a)* figured out that important ergonomic factors are the table's dimensions (height, width, depth), the radius of the curve and the backward inclination. After comprehending general ergonomic requirements, they conducted a user study using mock-up and paper prototyping where they tested several combinations of the different input factors resulting in 18 prototype alternatives *(see figure 6c and d)*. The test persons had to solve task on each option and rate them afterwards. Then, the three best prototypes had to be tested again to find out each participant's favored combination. Thus, the optimal values of the influencing factors could be determined.

Interestingly, the related venture named *BendDesk* [37], which has similar aims *(see figure 6b)*, uses different values concerning the vertical display's height and especially its inclination (actually none). In their paper, they claim that these were obtained by a preliminary user study, without giving further details about the procedure. However, the extensive testing within the *Curve* project showed that especially having a backward inclination, and thus not a fully vertical display, is essential for comfortable ergonomic working. A curve of 90 degrees, as being used in *BendDesk*, is much more tiring for the arms and harder to be driven along with a finger, as a tilted screen. Another difference between these two projects is the screen resolution: the *BendDesk* provides only 26 dpi[4] whereas the *Curve* will employ four high definition projectors[5] and thus will be able to attain a resolution of 81 dpi. For optimal ergonomic text reading, about 90 dpi are advisable (according to Ziefle [41])! Thus, reading text on the *BendDesk* will assumingly be a bother.

Unfortunately, the *BendDesk* team did not release any more information about their preceding study so far and thus it remains unclear, why they think a fully vertical display might be optimal.

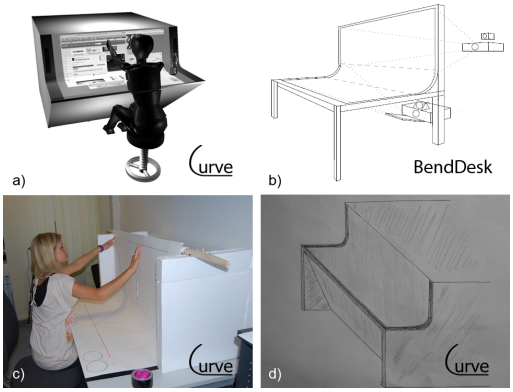


Fig. 6. *Curve* vs. *BendDesk*. *a)* Design of the *Curve* table [39]. *b)* Design of *BendDesk* [37]. *d)* Similar to *BendDesk*, the initial concept of *Curve* comprised also an angle of 90 degrees. *c)* The usability of that angle was later disproved in a extensive user study. *BendDesk* stuck to that value. (Images *c)* and *d)* have thankfully been provided by a *Curve* staff.)

4.1.6 Interactive System for Group Learning Support

In a project where user-centered design is mandatory, researchers built an interactive system, which focused on supporting group learning in elementary schools [33]. It deals with the physical construction of a town model, which is digitally augmented to give the children feedback to their constructions.

[3]Tangible User Interface.

[4]Dots Per Inch.

[5]currently two projectors are employed, four are planned for the future.

They started their design process with researching appropriate hardware as well as discussing with teachers, to gain content-related needs. Based upon this, the authors thought out the system's key facts and formed an initial idea. Instead of implementing their concept, they preferred building and evaluating a prototype to include user opinions. The board, where the model should be constructed on, was made of paper, as well as the objects themselves. The later system should identify the arrangement of the model but since this was a low-fidelity prototype, experimenters manually entered the particular locations (Wizard of Oz prototyping). Based on their observations, they could define their concept more precisely and learned important mistakes which were not to be made. After this, they implemented the first version of the system, evaluated it in a second user study and due to recommendations concerning the user interface, re-designed it afterwards.

Initially conducting a user study was highly advantageous as the project mainly targets children, whose behavior often differs from the one of an adult and thus problems could be revealed, which designers never have thought of.

4.2 Prototyping Interaction Techniques

The previous paragraph gave insights into the practical usage of prototyping in association with various types of interactive surfaces. This section deals with projects which focused on interaction techniques.

4.2.1 Whiteboard-based Interactions

A different way of approaching a development process, than the ones before, is by initially examine existing products and deduce significant errors from using them. The project idea arises afterwards, with these findings in mind. That way, Rekimoto began his work which dealt with a digital whiteboard and proper interaction techniques [29]. He and his scientists planed to support informal meetings with a digital whiteboard and started off with installing an existing whiteboard and a projector, and tested them using various emulation systems and GUI applications. They "immediately noticed a number of limitations that hamper effective collaborative activities" and then figured out the key problems: "Text entry is difficult", "Handling of existing data is problematic", "Large displays size makes current GUI design ineffective" and "Interactions with the whiteboard become a bottleneck" [29].

Based on these findings, they worked out a multi-device approach, where handheld devices are integrated in that scenario, which solve some of these issues, like text entry or handling existing data. The interaction is done via picking an object with a stylus on one display and releasing it on another. After the implementation, the researchers evaluated the system in a user study and got most widely positive feedback. However, one slight shortcoming became clear during the test: the weight of the palmtop device was too heavy.

They have developed a second version, where they claimed that the handheld has got lighter, but to avoid this mistake even already in the first generation design, they should have evaluated a prototype before the implementation, which could have had the final hardware parts, but only with paper prototypes for the screen's content. Aside from this issue, they did well to examine and evaluate existing products first, so that they were able to begin where others broke off.

4.2.2 ShapeTouch

Also *ShapeTouch* [7] started off from a point where real conditions were examined. They attended to fundamentally modify the way of interacting with an interactive surface, by abandoning the idea of using the finger as a mouse. Because direct manipulation of an object is possible, the means of interacting with a real object should be the template for designers instead, and not a desktop computer metaphor. Thus, they began their research by examining how we handle physical objects, like for example how we grab them, how we move them in the whole and how we perform tasks on them, from which they then deduced a set of important gestures *(see figure 7 for both, the pattern from the real world and the virtual gesture)*. Moreover, they noticed that the shape and size of the contact regions are important for the gesture determination and that they can profit from the fact that a lot of existing interactive surfaces already detect the shape of the area which is touching them.

They implemented their gesture recognizing algorithm and tested it on a tabletop, which sensed input via an infrared camera beneath. Thus, the shape of fingers, hands or objects laid on the surface could be determined. The software has been implemented relatively simple to allow rapid prototyping, in order to test and evaluate the basic handling of the interaction techniques together with users, without extremely stressing performance optimization. In a user study, the test subjects could freely explore the prototype for a few minutes, before they were given the basic instructions about the input methods bit by bit, including time to experiment around again. One of the findings was that the desktop metaphor was prevailing, rather than the way how they handle objects in reality. Thus, participants initially tried to control the objects like with a mouse, before they were given hints to try more realistic options. A problem which arose, was the lack of visual feedback about the virtual force, which virtually represents the pressure when touching a real object through the amount of fingers laid on a synthetic one. Also significant performance issues emerged, which were effects of the more rapid than rapidness-orientated implementation.

However, they say that it was favorable to evaluate a prototype with only basic functionalities, in order to get intermediate feedback which can be used to build further versions.

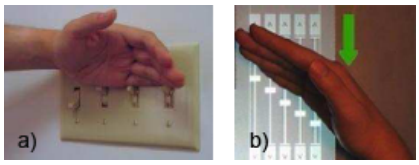


Fig. 7. *ShapeTouch* [7]. *a)* Pattern from the real world have been used as starting point to devise *b)* appropriate virtual counterparts.

4.2.3 Dynamo

A very long and extensive development process has been undertaken as *Dynamo* [16] was designed. Izadi et al. wanted to integrate interactive displays in large sociable spaces, like break rooms in schools or cafes in universities, to enable or trigger social interaction and support displaying and sharing of brought-in information. The total development lasted about three years and was divided into several phases.

The first one was concerned with ethnographic research about shared spaces, the behavior of people in them and how collaboration with personal devices takes places in there. This studies revealed the issue that mobile phone displays are too small to be apparent for a larger group and public screens, in contrast, are not able to support collaboration. Thus they derived the goal of enriching communal places with interactive displays to support mainly but not only collaboration and data exchange. Next, they conducted lab studies to explore the design space, as they call it. Using low-fidelity prototypes, they figured out the key design factors, namely whether single- or multi-user to be supported, the means of input and the physical orientation. They decided to use a vertical display (due to better visibility) and multi-user capabilities (because this supports a better collaboration). Moreover, a mouse- and keyboard-based instead of a pen-based interaction has been chosen, due to feelings of social embarrassment for a single person when having to stand in front of a crowd to reach the display with the pen. Each of these parameters were derived from a separate user study.

After these studies, a prototype has been implemented, including these mentioned features and additionally a device hub, were people can connect their own brought-in devices to, and thus are able to exchange or present their data with or to others. This prototype has been evaluated in a hotel's foyer and a conference center. Problems emerged during these tests like privacy and security concerns regarding the personal data and it became clear that some limitations in the usage are needed, to avoid occasional intrusions. Thus, *Dynamo* was altered

and the ways of interaction were refined. Subsequently, the developers wanted to evaluate their prototype in the longer run and in a real scenario. Thus, they decided to install it in a common room in a high school for ten days and observed how it was used and whether it was accepted.

This project shows a paradigmatic process of development. This can be linked to following aspects:

- Prototyping has been used, with a lower fidelity in the beginning, where the the final idea was uncertain and alternatives were existent, and an incremental raise during the phases,
- both main advantages of prototyping have been leveraged: prototypes as decision support within the research group as well as a method to gain user feedback, and
- at first, key factors of the design have been evaluated separately to avoid mutual influence, and after clarifying all parameters, the system as a whole has been tested.

4.2.4 Fighting for Control

Related to the mentioned project which dealt with group learning support, also the work of Marhsall et al. focuses on children's behavior when using a tabletop, though this time more emphasize is put on their way of interacting by itself and less on the system design [23]. In a study, participants (at about seven years of age) were asked to build a classroom seating plan and therefore were arranged in groups of 3 pupils. Some groups had to solve this task by using a paper prototype, where the objects (tables and the children's names) were small cardboard tokens, being able to be moved around and organized freely. Others should handle the work on an interactive tabletop (*DiamondTouch* [9]), where the items were available only in digital form. Additionally, one group should use both.

Besides the way how the children behave when solving the task, and not at least how they treat each other, the researchers also wanted to see whether there is a difference between the pupils' bearing when using the paper prototype or the tabletop. In fact, there were differences: because infants often fight over things like toys, they did the same when moving the tables or the names around to arrange a room plan. But a paper item can be drawn off and protected against others, whereas virtual objects could only been dragged within the display's boundaries. Thus, the children had to find other ways of limiting the access to an item, which resulted in physically pushing away the other person or at least his or her arm. This happened constantly during the sessions with the tabletop and in contrast, only one time with the paper prototype.

This result shows that when developing an interactive system where kids are involved, other factors come into play which affect the decision what prototype to take. Designers should be aware of this.

5 Discussion

The previous paragraphs showed that the integration of prototyping in the development process of interactive surfaces has many advantages. Some of these cover benefits within the design team, to reveal shortcomings on the one hand and new features on the other hand (like the technical interferences when testing *DiamondTouch* [9] or the new idea of designing objects to influence the sensing technology). They also simply uncover hardly predictable issues, like the dependency of the system's functionality on the time of day [19]. Other benefits are resulting from evaluating prototypes by users, because the way how they utilize a system can practically never be prognosticated certainly, especially when people are totally new to this are or when they are children [33]. Besides the retrieving of usability issues [29], participants of a user study can also be valuable when they are able to give hints for new design features [22].

However, some aspects which are to be heeded and even some disadvantages of prototyping, are still to be mentioned. For one thing, prototypes are likely to be overloaded with features, because often only a few user studies are conducted and thus as much as possible is tried to be evaluated in them. This leads to an overload of the test person and thus in total, less is figured out. A better approach has been made with *Hermes* [11], where a lot of short and iterative prototyping cycles have been used and each time, only few things have been changed. Thus, more qualitative results were the outcome, however suffering losses in time. This can be mentioned as another problem: When applying prototyping wrong, it can cost a lot of time while producing no benefit. This challenge is further aggravated, as the process of prototyping is often not easy to manage and thus a lot previous knowledge is needed to integrate it efficiently. Another drawback is that the level of detail is often limited either in vertical or horizontal direction. When designers fail to clarify this, the feedback of test subjects might be influenced negatively by the lacking details.

As one main goal of prototyping is to reveal possible deficiencies as soon as possible, for some projects the question arises, whether they could have prevented later failings if they would have applied prototyping. This question is even more interesting, as this work deals with interactive surfaces, where subsequent adjustments can be extremely expensive, because a lot of high-priced hardware is used. The *BendDesk* [37] is such a project, though there are no known problems yet. It is conceivable that they might suffer from insufficient prototyping afterwards, if the *Curve* [39] researchers are right and an angle of 75 degrees is the absolute maximum for the vertical display (compared to 90 degrees in the case of *BendDesk*). In other research cases, prototypes are not built for an iterative or user-centered design process, but only for a final evaluation of the project [18, 12, 30, 24]. To stick to the prior question, *EnhancedTable* [18] is another example: a CSCW[6] system was built to augment group meetings by enabling participators to share and view documents on a table. The interaction is done by using the fingers for direct touch - the most common input method on an interactive surface. However, the user study they conducted after their implementation showed that the people did not want to use this common way of input but rather one which is pen-based - because during a meeting, they usually hold a pen in their hands *(see figure 8)*. By integrating a study earlier in the development process, this issue had been detected in due course of time and could have been considered during their implementation.

However, it must be remarked that it is often hard to make a clear statement about whether and how particular projects have used prototyping, because in many cases, the whole design process is not really laid bare in their reports - as barely as subsequent mistakes are.

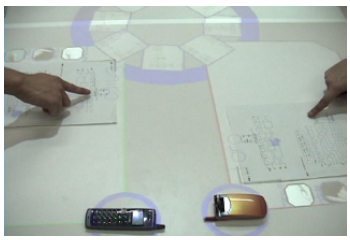


Fig. 8. *EnhancedTable* [18]. Interaction is done via the common way of directly touching the surface with a finger. However, a final user study revealed that a pen-based input would have been the favored method.

6 Conclusion

This work delved into exploring how prototyping was used and can be used when designing interactive surfaces and corresponding interaction techniques. The subject *prototyping* was regarded from a general point of view, together with a classification for this specific topic *interactive surfaces*. Various related prototyping tools were introduced, each one targeting different purposes within the subject. Moreover, recent research projects were presented, dealing with the design of interactive surfaces or special techniques for interacting with them. In summary, it can be stated that prototyping supports effective collaboration within development groups and takes user needs into account of the design process.

[6] Computer Supported Cooperative Work.

Today, only 24%[7] would use a tabletop instead of a standard pc (as claimed in a survey), mostly due to lacking means of text entry, standard applications and ergonomics [15]. This shows that the current state-of-the-art still has enough to improve, especially in terms of usability, whereby the use of prototyping can be of great value. In addition, a lot of research focuses on multi-touch but completely disregards proper and ergonomic means for text entry. So, broken down to this survey, would you trade multiple mice for your desktop keyboard?

REFERENCES

[1] Apple - iPhone, http://www.apple.com/iphone/, requested on: 2010-01-10.

[2] Microsoft - Surface, http://www.microsoft.com/surface/, requested on: 2010-01-10.

[3] The OpenInterface Project, http://www.oi-project.org/, requested on: 2010-01-10.

[4] Nintendo - Wii, http://www.nintendo.com/wii/, requested on: 2010-01-10.

[5] Definition of prototyping from the online dictionary "dictionary.com". http://dictionary.reference.com/browse/prototyping, requested on: 2010-01-12.

[6] E. Akaoka and R. Vertegaal. Displayobjects: functional prototyping on real objects. In D. R. O. Jr., R. B. Arthur, K. Hinckley, M. R. Morris, S. E. Hudson, and S. Greenberg, editors, *CHI Extended Abstracts*, pages 3507–3508. ACM, 2009.

[7] X. Cao, A. D. Wilson, R. Balakrishnan, K. Hinckley, and S. E. Hudson. Shapetouch: Leveraging contact shape on interactive surfaces. In *Tabletop*, pages 129–136. IEEE, 2008.

[8] J. Davis and X. Chen. Lumipoint: multi-user laser-based interaction on large tiled displays. *Displays*, 23(5):205 – 211, 2002.

[9] P. H. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *UIST*, pages 219–226, 2001.

[10] A. J. Dix, J. Finley, G. D. Abowd, and R. Beale. *Human-Computer Interaction (3rd ed.)*. Prentice Hall, New York, 2004.

[11] D. Fitton, K. Cheverst, C. Kray, A. Dix, M. Rouncefield, and G. Saslis-Lagoudakis. Rapid prototyping and user-centered design of interactive display-based systems. *IEEE Pervasive Computing*, 4(4):58–66, 2005.

[12] T. Gross, M. Fetter, and S. Liebsch. The cuetable: cooperative and competitive multi-touch interaction on a tabletop. In M. Czerwinski, A. M. Lund, and D. S. Tan, editors, *CHI Extended Abstracts*, pages 3465–3470. ACM, 2008.

[13] B. Hartmann, S. R. Klemmer, M. Bernstein, L. Abdulla, B. Burr, A. Robinson-Mosher, and J. Gee. Reflective physical prototyping through integrated design, test, and analysis. In P. Wellner and K. Hinckley, editors, *UIST*, pages 299–308. ACM, 2006.

[14] O. Hilliges, D. Kim, and S. Izadi. Creating malleable interactive surfaces using liquid displacement sensing. In *Tabletop*, pages 157–160. IEEE, 2008.

[15] A. B. B. A. D. W. Hrvoje Benko, Meredith Ringel Morris. Insights on interactive tabletops: A survey of researchers and developers. 2009.

[16] S. Izadi, G. Fitzpatrick, T. Rodden, H. Brignull, Y. Rogers, and S. Lindley. The iterative design and study of a large display for shared and sociable spaces. In *Proceedings of the 2005 conference on Designing for User eXperience*, 2005.

[17] H. Koike and M. Kobayashi. Enhanceddesk: Integrating paper documents and digital documents. In *APCHI*, pages 57–62. IEEE Computer Society, 1998.

[18] H. Koike, S. Nagashima, Y. Nakanishi, and Y. Sato. Enhancedtable: Supporting a small meeting in ubiquitous and augmented environment. In K. Aizawa, Y. Nakamura, and S. Satoh, editors, *PCM (1)*, volume 3331 of *Lecture Notes in Computer Science*, pages 97–104. Springer, 2004.

[19] H. Koike, Y. Sato, and Y. Kobayashi. Integrating paper and digital information on enhanceddesk: a method for realtime finger tracking on an augmented desk system. *ACM Trans. Comput.-Hum. Interact.*, 8(4):307–322, 2001.

[20] M. Kranz and A. Schmidt. Prototyping smart objects for ubiquitous computing. In *In Proceedings of the International Workshop on Smart Object Systems in Conjunction with the Seventh International Conference on Ubiquitous Computing*, September 2005.

[7] of 58 interviewees, most of them involved in tabletop research and thus not foreign to the topic. Status: March 2009.

[21] Y.-K. Lim, E. Stolterman, and J. Tenenberg. The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. *ACM Trans. Comput.-Hum. Interact.*, 15(2):1–27, 2008.

[22] V. Maquil, T. Psik, and I. Wagner. The colortable: a design story. In A. Schmidt, H. Gellersen, E. van den Hoven, A. Mazalek, P. Holleis, and N. Villar, editors, *Tangible and Embedded Interaction*, pages 97–104. ACM, 2008.

[23] P. Marshall, R. Fleck, A. Harris, J. Rick, E. Hornecker, Y. Rogers, N. Yuill, and N. S. Dalton. Fighting for control: children's embodied interactions when using physical and digital representations. In D. R. O. Jr., R. B. Arthur, K. Hinckley, M. R. Morris, S. E. Hudson, and S. Greenberg, editors, *CHI*, pages 2149–2152. ACM, 2009.

[24] M. Matsushita, M. Iida, T. Ohguro, Y. Shirai, Y. Kakehi, and T. Naemura. Lumisight table: a face-to-face collaboration support system that optimizes direction of projected information to each stakeholder. In J. D. Herbsleb and G. M. Olson, editors, *CSCW*, pages 274–283. ACM, 2004.

[25] M. R. McGee-Lennon, A. Ramsay, D. McGookin, and P. Gray. User evaluation of oide: a rapid prototyping platform for multimodal interaction. In *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 237–242, New York, NY, USA, 2009. ACM.

[26] M. R. Morris, A. J. B. Brush, and B. Meyers. A field study of knowledge workers' use of interactive horizontal displays. In *Tabletop*, pages 105–112. IEEE, 2008.

[27] T.-J. Nam. Collaborative design prototyping tool for hardware software integrated information appliances. In R. Shumaker, editor, *HCI (14)*, volume 4563 of *Lecture Notes in Computer Science*, pages 504–513. Springer, 2007.

[28] J. Patten, H. Ishii, J. Hines, and G. Pangaro. Sensetable: a wireless object tracking platform for tangible user interfaces. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 253–260, New York, NY, USA, 2001. ACM.

[29] J. Rekimoto. A multiple device approach for supporting whiteboard-based interactions. In *CHI*, pages 344–351, 1998.

[30] J. Rekimoto and M. Saitoh. Augmented surfaces: A spatially continuous work space for hybrid computing environments. In *CHI*, pages 378–385, 1999.

[31] T. Sato, H. Mamiya, H. Koike, and K. Fukuchi. Photoelastictouch: transparent rubbery tangible interface using an lcd and photoelasticity. In A. D. Wilson and F. Guimbretiere, editors, *UIST*, pages 43–50. ACM, 2009.

[32] G. Shoemaker, A. Tang, and K. S. Booth. Shadow reaching: a new perspective on interaction for large displays. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 53–56, New York, NY, USA, 2007. ACM.

[33] M. Sugimoto, F. Kusunoki, and H. Hashizume. Design of an interactive system for group learning support. In *Symposium on Designing Interactive Systems*, pages 50–55, 2002.

[34] B. T. Tognazzini. The "starfire" video prototype project: a case history. In B. Adelson, S. T. Dumais, and J. S. Olson, editors, *CHI*, pages 99–105. ACM, 1994.

[35] J. Underkoffler and H. Ishii. Urp: a luminous-tangible workbench for urban planning and design. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 386–393, New York, NY, USA, 1999. ACM.

[36] D. Vogel and R. Balakrishnan. Interactive public ambient displays: transitioning from implicit to explicit, public to personal, interaction with multiple users. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 137–146, New York, NY, USA, 2004. ACM.

[37] M. Weiss, S. Voelker, and J. Borchers. Benddesk: Seamless integration of horizontal and vertical multi-touch surfaces in desk environments. 2009.

[38] P. Wellner. Interacting with paper on the digitaldesk. *Commun. ACM*, 36(7):87–96, 1993.

[39] R. Wimmer, F. Schulz, F. Hennecke, S. Boring, and H. Hußmann. Curve: Blending horizontal and vertical interactive surfaces. 2009.

[40] J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: a 1 recognizer for user interface prototypes. In C. Shen, R. J. K. Jacob, and R. Balakrishnan, editors, *UIST*, pages 159–168. ACM, 2007.

[41] M. Ziefle. Effects of display resolution on visual performance. *Human Factors*, 40(4):554–555, 1998.

Prototyping of Interactive Surfaces for mixed Physical and Graphical Interactions

Anna Tuchina

Abstract—In this work I examine several possibilities for the prototyping of interactive surfaces for mixed physical and graphical interactions. These interactive surfaces belong to the so called tangible user interfaces, which play an elemenatry role in ubiquitous computing. So far there is no universal answer as to what prototyping technique is the most appropriate for the development of these interfaces. Low and high fidelity prototyping are introduced as two different approaches, which both can be useful in this case. In recent scientific publications several prototypes which explore the possibilities of these interactive surfaces have been described. I discuss the prototypes and examine their individual advantages and disadvantages. I then identify a shift to high fidelity prototyping as a central trend in this research, and finally discuss how low and high fidelity prototyping can be reasonably combined.

Index Terms—Ubiquitous Computing, Interactive Surfaces, Tangible User Interfaces, Hi-fi Prototyping, Lo-fi Prototyping, Interactive Prototyping, Paper Prototyping

1 Introduction

Since the beginning of the computer era three waves of computing can be identified. In the first one many people shared one computer. This so called mainframe wave was followed by the personal computer wave, where each person uses one computer. Since the 1990s the third wave, ubiquitous computing, began to develop, where each person uses many computers [16].

But even more than that, ubiquitous computing stands for invisible technology, where users interact with common objects without seeing the technology behind it [15]. While in the first two waves the people had to deal with the technology in order to use it, the ubiquitous computing paradigm is aimed at completely adapting the computer to the world of the user. Users then can interact with physical objects wherein computers are embedded, but no longer have to be concerned about technical details.

To make the ubicomp vision reality, new user interfaces must be developed as an alternative to the classical desktop PC interface. Such an alternative are the interactive surfaces. Interactive surfaces which combine physical and graphical interactions belong to the so called tangible user interfaces (TUIs). TUIs form a counterpart to the well known graphical user interfaces (GUIs) *(see figure 1)*.

As opposed to the GUI a TUI allows direct manipulation of digital objects through physical objects. That allows interactions, which are impossible to accomplish with a classical mouse and keyboard, like two-handed and collaborative interaction and simultaneous alteration of position and orientation. Furthermore, a TUI is designed to be the ideal interface for one specific task, while a GUI uses one interface to accomplish different tasks and therefore cannot offer the best possible interface for each application. Using TUIs takes advantage of a humans natural capability for sensing and manipulating his physical environment and his spatial reasoning skills [1]. All in all, TUIs are designed to support human interaction skills.

2 Prototyping Techniques

Since the research on TUIs is in an early stage, it is unclear which prototyping techniques are most suitable for the development process. There are some but few studies which concentrate on this topic [10] [5]. They compare paper prototyping which is often used in GUI design to interactive prototyping and its use for ubiquitous computing.

- *Anna Tuchina is studying Media Informatics at the University of Munich, Germany, E-mail: toutchina@cip.ifi.lmu.de*

- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009*

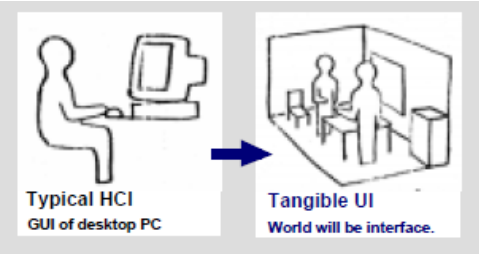


Fig. 1. GUI and TUI in comparison [1]

Paper prototyping is a low fidelity, interactive prototyping a high fidelity prototyping technique [9]. Both methods are briefly introduced in the following sections.

2.1 Low Fidelity Prototyping

Low fidelity Prototypes, for example paper prototypes, are often used in the early design process. They are cheap and easy to create and can be altered very quickly. They offer limited or no functionality and cannot respond to user input automatically. Therefore the interaction with a low fidelity prototype is scripted and supported by a facilitator. Lo-fi prototypes can be used to run usability tests. Due to the quick alteration, it is even possible to adjust the prototype during a test session. The prototype is made quickly without paying attention to details and therefore is convenient to test general concepts without being distracted by particular elements. On the other side it is problematic testing interactions in a realistic way. First, the user can see what the facilitator is doing and therefore has information that normally would be hidden from him. Second, the interaction of the user is interrupted by the delay that the facilitator produces during updates. Furthermore, user studys that are conducted using a lo-fi prototype usually are time-consuming and require a lot of manpower.

2.2 High Fidelity Prototyping

High fidelity prototypes are fully interactive. They represent the core functionality of the future product. Development of hi-fi prototypes usually takes a lot of time, but in return offers a realistic interface. Interactive prototypes can easily be used for user tests. The interaction with the hi-fi prototype happens in a realistic way. During the test little or no staff is needed. High fidelity prototypes are especially useful to test the user interface, because the look and feel is very similar to the final product. Creating and altering an interactive prototype involves much more time, cost and expert knowledge. Therefore it is not efficient to test different concepts or identify requirements.

Obviously both prototyping techniques have their advantages and

disadvantages. The interesting question is how to use them appropriately.

3 Related Work

In the following part of this paper different prototypes of interactive surfaces involving tangible user interfaces will be discussed. Special interest lies in the construction and evaluation of the prototype, i.e. how time-consuming and complex was the construction of the prototype, and what findings have been provided using it. Finally it will be discussed if better insights could have been gained by a different approach.

3.1 Tangible Geospace

In their research about tangible bits Hiroshi Ishii and Brygg Ulmer present the metaDesk as one of their ubiquitous computing prototype [1] [11]. Based upon the idea of foreground and background bits the metaDesk and the transBoard were constructed to research direct interaction with tangible objects, while the ambientRoom examplifies peripheral communication.

3.1.1 metaDesk

Characteristic for the metaDesk is the idea to represent classical GUI elements with physical objects *(see figure 2)*. For example a TUI "lens" represents the window and the TUI "phicon" is literally a physical icon.

The setup of the metaDesk is shown in figure 3. With one projector for the 2D view of the table surface, one dispay for the 3D view of the active lens, a number of physical objects, a computer-vision system with several sensors and three computers this prototype was very intensive in terms of hardware.

On the metaDesk platform an application called tangible geospace was installed. This application allows the user to place phicons representing buldings of the MIT campus on the metaDesk, where a map of the campus appears. The map is positioned in a way, that makes the phicon stand on the spot on the map where the building it is representing is located. The active lens delivers the matching 3D picture. By adding a second phicon onto the desk, the map will be scaled, rotaded and translated automatically, so that both phicons stand on the correct positions.

The application is supported by an extensive software, that was entirely implemented by the authors. The time that was needed in order to build this prototype is not mentioned in the available sources. But due to the both complex hardware and software components, it can be assumed that constructing it took probably several weeks up to months.

Fig. 2. User Interface Elements of the metaDesk [11]

3.1.2 Discussion

The metaDesk was developed with two intentions. First it is a proof-of-concept setup for the desktop metaphor and second it was built to help evolving new interaction techniques. The question is if the metaDesk was appropriate to deliver it. As a high fidelity prototype, it took a very long time in development, and was neither useful to test the concept, nor to compare different concept approaches. No user studys or experiments using metaDesk were mentioned in the papers. It was used more as a showcase for visitors or passers-by to interact with. In fact it seems as if more insights into technical details were gained during the development than into new interaction techniques.

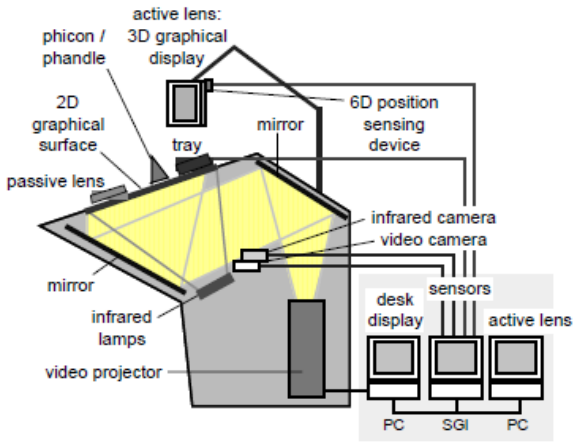


Fig. 3. Architecture of the metaDesk [11]

3.2 Urban Simulation

The urban simulation was first realised by Hiroshi Ishii and John Underkoffler with the uban planning workbench called Urp [14]. Many concepts that were used in Urp were developed in the previos prototype called Illuminating Light [12] [13].

3.2.1 Illuminating Light

The interface of Illuminating Light serves the purpose of simulating laser light that interacts with different optical elements and is a toolkit for holographic recording setups. The laser beam is simulated by light projected on a table. In addition the optical elements needed for a holographic recording setup are represented by simple physical objects. After placing a "laser" on the table a beam of light is projected where a laser light would normally be. By placing "optical elements" on the table the projected beam of light can be modified and a holographic 2D image of a physical 3D object occurs *(see figure 4)*.

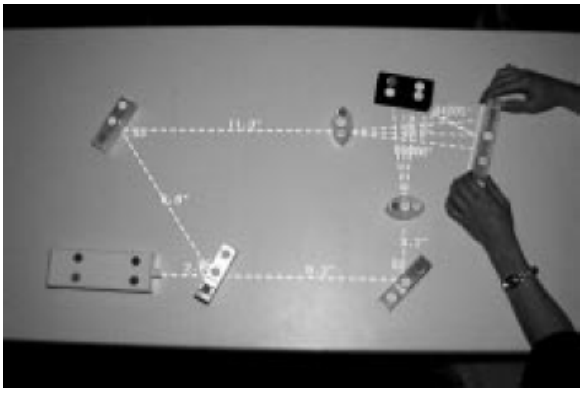


Fig. 4. Holographic recording setup with the Illuminating Light toolkit [12]

For this scenario Ishii and Underkoffler created a high fidelity prototype. This prototype consists of plastic forms as representations of the optical instruments and the laser, which are tagged for identification with colored dots *(see figure 5)*. While technically the shape of

all objects could be the same, the authors consider design will be an important element in the final interface. Therefore the physical objects assume the shape of the real things they are representing.

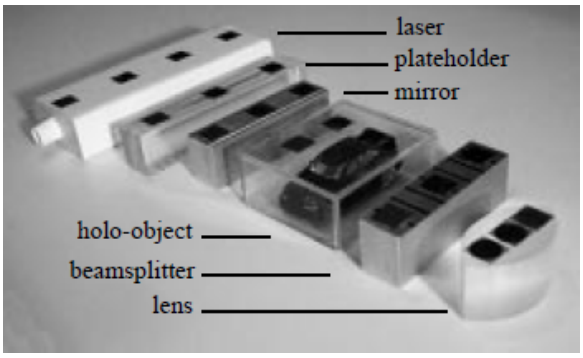


Fig. 5. Representations of the optical objects [12]

Another part of the prototype is a construction comprising a camera and a projector, the so called I/O Bulb, which provides input through the camera to the system in the background and output through the projector on to the table at the same time *(see figure 6)*. The system in the background handles the input from the I/O Bulb in a pipeline with four parts:

- glimpser: gets images from the I/O Bulb and identifies objects by their dot pattern
- voodoo: analyzes the patterns and detects position and orientation of the objects
- simulator: determines the path of the laser light
- rendering: renders the laser light and sends it to the I/O Bulb

Trying to keep the time needed to create this prototype within limits, Ishii and Underkoffler are using available software toolkits like voodoo, a geometric parsing toolkit, and the glimpser, which was evolved from an earlier version of a program built by the authors. The whole construction took approximately one week.

Using the illuminating light prototype a user study with eight persons was conducted. Most users found the interface believable and realistic after they got used to it. Many comments were made about the graphic presentation of information, e.g. about the presentation of distances. The two dimensional view of the hologram was criticised, given that the third dimension also contained important information.

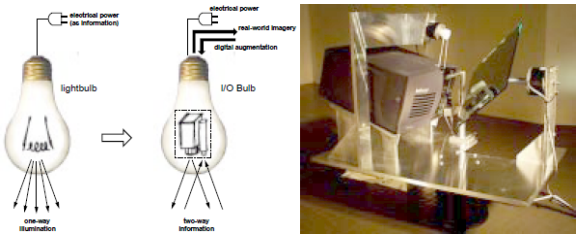


Fig. 6. I/O Bulb in theory and practise [12]

3.2.2 Urp

The Urp system is an interface for urban planners that allows them to work on both the aesthetic and practical aspects of design. For that pupose the following information is needed:

- shadows cast by buildings
- proximities between buildings and other elements like streets

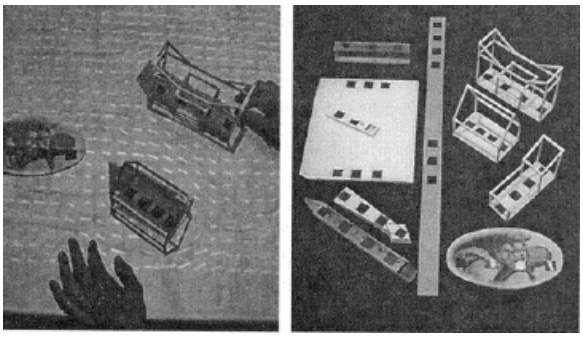


Fig. 7. Physical objects and the wind flow simulation used in Urp [13]

- reflections from buildings that have glass sides
- wind flow and its influence on buldings
- visual space

The Urp prototype is based upon Illuminating Light. The I/O Bulb and physical objects with colored dots were also used for this setup. For recognition of the physical objects again the pipeline with glimpser and voodoo is used. In order to simulate wind another software has been added, the cellular automaton called "lattice gas".

As the physical objects, except for the buldings, cannot be assigned with a shape from real things, there is no specific design by what the user could recognise their function *(see figure 7)*.

No formal studies were conducted using Urp, but there was feedback given by professional urban planners and visitors to the lab. The Urp interface was accepted very well. As a main application area the professionals identified the presentation for clients. Problems were discovered in the visual space and wind flow functions. Urban planners normally need a site view from the entrance of a bulding. While this is easy to perform in virtual 3D space, it is impossible using the Urp interface, given that the scale of the camera and the buildings in Urp do not match the real world. Also the wind flow is not represented in a realistic way with a 2D projection.

3.2.3 Discussion

Both prototypes faced the same two main problems. First of all the design of the physical objects was recognised as important but has not been investigated in a separate study. This could have been easily done using paper or plasticine. While it didn't seem to be an issue in the Illuminating Light prototype it definitely became one in Urp. While Illuminating Light objects were shaped like the optical objects they were representing, this was not possible for the Urp objects.

The second problem also appeared in both prototypes, where 3D phenomenons were represented by 2D views. In Illuminating Light it was the hologram and in Urp the wind flow. Both effects were lacking the information they have in reality. The alterations that need to be done to the prototypes to solve this problem could interfere with the whole design. When using a high fidelity prototype, changing parts of it takes a lot of time. With low fidelity prototypes this would have been much easier and if the whole concept turns out to be inappropriate, lo-fi prototyping allows to develop a new concept much quicker.

3.3 From Sensetable to Audiopad

While Sensetable was not developed for a specific application but more for general research purpose [6], Audiopad was designed in order to be the optimum interface for musical performance [7] [8]. However Audiopad could take advantage of the findings that were made with the Sensetable platform.

3.3.1 Sensetable

The platform Sensetable is not built upon a specific scenario. The authors from MIT Media Lab saw the purpose of Sensetable in the

research of general technical as well as conceptual principles in TUI design. The technical goal was to be able to track six to ten objects on a table surface with low latency. Conceptually the aim was to make solid tangible objects changeable in adding another elements to it and so compensate the disadvantage physical objects have compared to graphical ones.

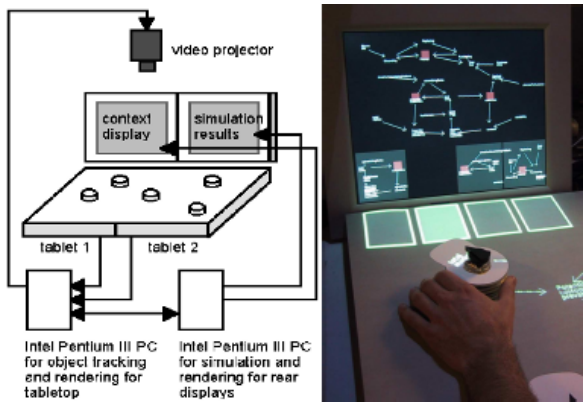


Fig. 8. Archtecture and User Interface of the Sensetable Platform [6]

To do that a high fidelity prototype has been created. For the construction of this platform two Wacom sensing tablets were composed to one interaction surface *(see figure 8)*. With the so created surface six pucks, made out of adjusted Wacom mice, were used as object representations. Since the Wacom tablets could only track two objects at a time, the authors wrote an algorithm that tracked the six objects in random turns, so that each object was tracked 30 percent of the time. In order to decrease the latency of a puck that is currently in use, sensors were added to the pucks. These sensors "turned on" the latency of the puck that was touched to 100 percent. As mentioned before the designers wanted the pucks to be alterable. Therefore the so called dials and modifiers can be adjusted on top of a puck. Using the Sensetable platform the following interaction techniques were observed in experiments:

- binding and unbinding of pucks to and from digital objects
- manipulating digital objects with pucks
- visualizing complex information structures
- sharing information between the tabletop and a display screen

In order to bind a digital object to a puck the user has to place the puck near this object. This principle is adequate for a small amount of information shown on the table, but with a great number of objects can cause difficulties in binding the correct object. For that reason the authors extend this principle by two properties: The distance between the digital objects is dynamically adjusted and the puck has to linger on the object for a certain time in order to bind it. Objects are unbound from the pucks by shaking them. Many users tried to unbind the puck by taking it off the surface. The authors intend to allow this possibility in their next prototype.

Also, users didn't really see the pucks as representations of the digital objects but only as control elements. To emphasise the representational character of the pucks, information about the object that was currently bound to the puck was now projected directly on the puck instead like before on the table surface near the puck.

Changes made to objects by dials or modifiers were projected on a screen in the beginning. The experiment revealed that the users wanted this information to be displayed near the object on the table surface and also wanted graphical feedback while changing it.

Whenever users wanted to interact with a great amount of information it was difficult for them to keep an overview. Therefore a layout algorithm was developed, that highlights important and weakens unimpotant information. What information is important depends on the particular application.

3.3.2 Audiopad

The next prototype called Audiopad is an interface for musical performance using the insights that were made through the development of the Sensetable platform. It contents smaller pucks that are tracked using RF tags and one sensing surface instead of two joined ones.

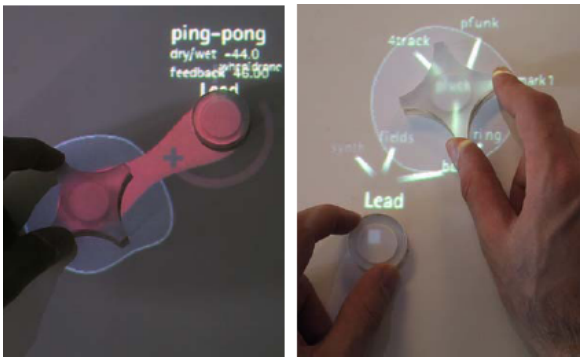


Fig. 9. Changing two dimensional parameters and browsing through the samples tree [8]

Also it is possible to determine the orientation of a puck which now contains two RF tags. The pucks now represent groups of samples. A second type of puck was added. This puck is called the selector puck and can be used to navigate through a tree of samples as shown in figure 9. This encourages two handed navigation. While the left hand is holding the samples puck the other hand is browsing through the samples tree using the selector puck.

Pucks can be rotated to change the volume of the selected track. On top of the sample puck a button is placed. Pressing the button activates the effect setting view of the track, so that these settings can be modified. In a later version of Audiopad this button is removed.

A floating menu is used to select items from it *(see figure 10 on the left)*. The user can move the puck over the items and select it by resting upon one of them. Moving the puck also moves the menu along with it to ensure that the menu assigned to the puck is always where the puck is. In experiments problems occured when the user accidentally selected an item from the menu although he only wanted to move the puck to a different position or when the user wanted to select an item but couldn't because the menu was moving. To solve this problem the so called selection area has been defined in which items can be selected. But this solution didn't really solve the problem. When users wanted to select an item they often made a loop with the puck that went outside of the selection area and this way was interpreted as a puck movement which also made the menu move. Therefore a different approach has been chosen, where the menu would be moved only if the puck stays out of the selection area.

Because of the so called "nulling problem" that came along with rotating the puck to change the volume, a different approach was chosen to manipulate this parameter. A so called master puck was added. The volume of the track now depends on the distance of the according puck to the master puck. The closer the puck is to the master puck the louder is the track that it represents *(see figure 10 on the right)*.

Another problem occured in connection with two dimensinal parameter changes. Different approaches have been tested. The first technique was to define absolute zones on the table where parameters could be modified along the x and the y axis. This approach implied the disadvantage of reducing the space that pucks could use on the table. A better solution employs a technique, where relative distances are used. In order to change settings the modifier puck is placed on a specific spot near the puck, then a barbell-formed area occurs enclosing the modifier and the sample puck as shown in figure 9. This graphical feedback shows the user how he is changing the settings compared to the starting position.

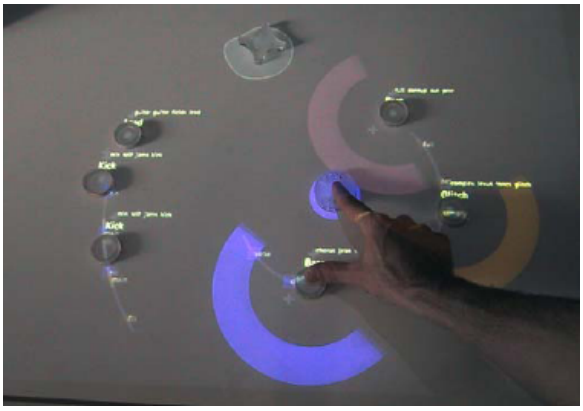


Fig. 10. The final interface of Audiopad [8]

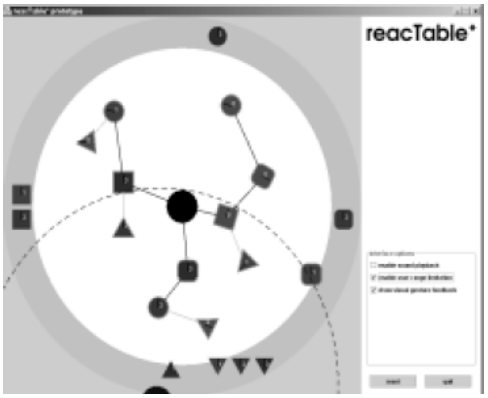


Fig. 11. reacTable simulation for PC [2]

3.3.3 Discussion

Using the Audiopad prototype many design and interaction issues have been discovered. Although the problem is that, implementing some of these changes could cause the necessity of constructing a new prototype. Also many problems have been recognized very late in the design process. It is possible that these interaction issues could have been exposed using a low fidelity prototype. This would have avoided time consuming construction and alteration of the high fidelity prototype. Especially in the early design phase the forms of the pucks and the general layout could have been easily tested with a low fidelity prototype.

3.4 Collaborative Musical Instrument

Sergi Jordà from the Music Technology Group in Barcelona developed the reacTable, a modern musical instrument, that uses the advantages of a tangible user interface [4] [3]. The main goal was to make the instrument easy to use even for beginners, but at the same time offer creative freedom and control possibilities for professional players.

3.4.1 FMOL

The history of the reacTable begins with a software application called FMOL, that was created by Jordà in 1997 [2]. FMOL is a tool for collaborative musical composition on the internet. It was built for PC and was controlled by mouse. The GUI consisted of a 6x6 grid, where the vertical lines represent voice generators and horizontal lines effect processors. The lines act as input as well as output. By editing the lines with the mouse, the wave form of this line changes. These lines can later also be found in the reacTable interface.

3.4.2 reacTable

The reacTable was developed to remove the restrictions in the interaction that were present in FMOL. Internet collaboration in FMOL was replaced by direct collaboration on a table.

In order to support the development of reacTable a simulation program for PC was prepared *(see figure 11)*. Using this simulation basic concepts and interaction ideas were tested before they were realised with a more complex prototype. Four kinds of objects were identified and assigned each with a different geometrical shape. Later the reacTable will contain six object families. Also, the connection lines in the final interface will embody additional information, while in the simulation they were just straight lines.

The final prototype is constructed with a round table, a camera and a projector, the tracking system reacTIVision, a connection manager to calculate the network and the audio- and the visual synthesizers *(see figure 12)*. Furthermore, there are pucks that have one of the six possible shapes depending on their content. On their bottom side markers were places in order to be able to track the pucks. On the upper side symbols specify the content.

The user can move the pucks auround on the table and thus create and edit a network. Whether pucks connect or not depends on the type of the pucks and their distance from each other. A connection can be ”muted” by a simple finger gesture. Editing object parameters can be executed by rotating the puck and by pointing on the round graphical elements that enclose the pucks.

The reacTable can be launched in different scenarios. Local collaboration on a reacTable can be performed by up to four persons at the same time. In doing so users can each play in a separate area, like it would be playing in a band, or they can construct one musical formation together *(see figure 13)*. Beside the local there also exists the remote collaboration, where reacTables or simulations of it are connected in a peer-to-peer network. Thereby objects that are controlled on the remote table also can be seen on one’s own table. But these objects are virtual and can only be controled on the remote table. Further scenarios are imaginable on mobile devices, where the reacTable is implemented in a reduced way.

The reacTable was showed at several occasions like festivals, conferences and shows, where it was played by many users with different musical knowledge.

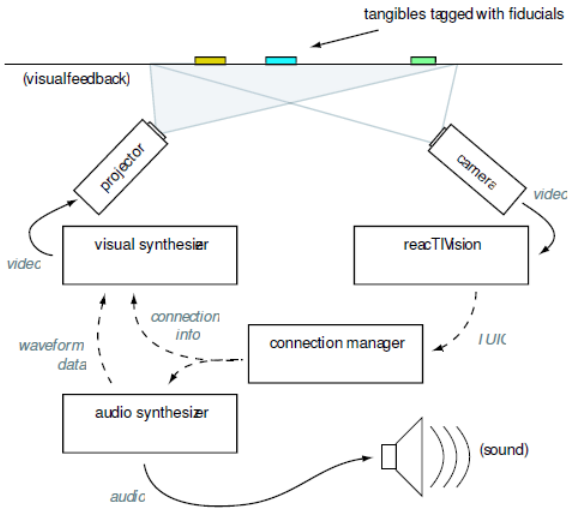


Fig. 12. Architecture of the reacTable [3]

3.4.3 Discussion

Using the reacTable simulation the designers had many possibilities to test, and if necessary change their concept without too high effort. It would also have been possible to use low fidelity prototyping techniques like paper prototyping in order to experiment with TUI

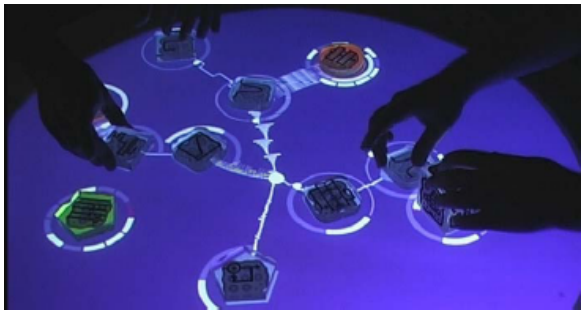


Fig. 13. The reacTable in collaborative use [3]

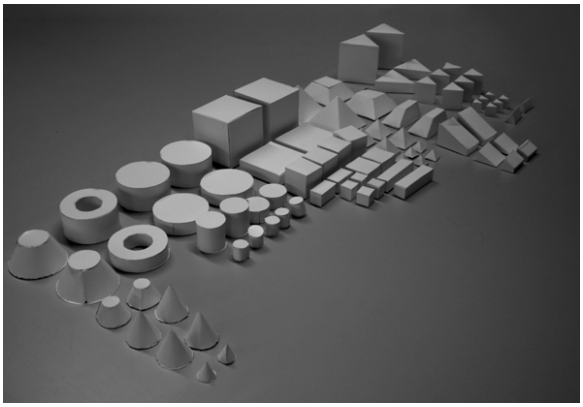


Fig. 14. TUI Elements in Paperbox3D

elements, which is even more beneficial concerning complexity and freedom of design.

4 Conclusion

All the setups that had started off as high fidelity prototypes, had to be replaced by new ones in order to advance. Especially at the beginning of the development, the focus often lies on technical implementation details and not on conceptual advancement. One could argue that technical progress is as important as is the concept for the final product. This is supported by the fact that a concept standing alone without any possible implementation doesn't have much value. Then again, a concept should not be constrained by the current technical possibilities, or it will be at risk to have the only purpose to fit the given technology. When the used technology becomes obsolete, so does the whole concept.

As we have seen, high fidelity prototypes are not suitable for concept development. They are inefficient in time and cost, this makes them unflexible to possible major changes to the concept. On the other hand, because of their resemblance to the final product, high fidelity prototypes can be useful to analyse interactions in a realistic way. In order to test different concept ideas low fidelity prototypes can be of a greater use. They can be quickly adapted if necessary. But it isn't possible to investigate interaction techniques in a realistic way using lo-fi prototyping. As this is an important field of current research on TUIs, low fidelity prototyping alone won't be enough.

Therefore it is advisable to use lo-fi prototypes especially in the early phase of the concept development in order to give the designer as much freedom as possible to alter it without having to think about platform constraints. Also it is possible that lo-fi solutions like paper prototyping not only make it easier for the users to criticise the concept, but also for the developers to deal with it. They then don't have to fear major changes in the concept because of the amount of work it could involve.

Later in the design process, where details need to be refined, hi-fi prototyping can build upon the lo-fi prototype. Little changes especially in the look, position and behaviour of graphical objects can be carried out easily. Great conceptual changes are not very likely to occur after testing the concept with the low fidelity prototype.

5 Outlook

As an alternative to high fidelity prototyping I want to introduce a paper prototyping technique that was especially developed for tangible user interfaces. The Paperbox 3D toolkit is currently being developed by Julia Küfner, Alexander Wiethoff and Andreas Butz from the media informatics department for computer science of the LMU. This paper prototyping toolkit has all the advantages that are associated with common paper prototypes. In addition they enhance two dimensional paper prototypes by offering the 3D shapes that are needed for TUI development as seen in figure 14.

Using this toolkit a study has been conducted with three different settings. In each setting the test persons had to use a different brainstorming method to develop a photobrowsing application on a tangible user interface. The groups using post-its or the clustering technique both have structured and analysed the topic in a more general way, but had little precise ideas. The group that used the Paperbox 3D toolkit had more specific ideas, but didn't concentrate so much on the overview.

For further development of this toolkit a major experiment and a user study will be conducted. Also a comparative test with a high fidelity prototype that has been constructed by members of the institute is planned.

References

[1] H. Ishii and B. Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, New York, NY, USA, 1997. ACM Press.

[2] S. Jordà. Sonigraphical instruments: from fmol to the reactable. pages 70–76, 2003.

[3] S. Jordà, G. Geiger, M. Alonso, and M. Kaltenbrunner. The reactable: exploring the synergy between live music performance and tabletop tangible interfaces. pages 139–146, 2007.

[4] M. Kaltenbrunner, S. Jordà, G. Geiger, and M. Alonso. The reactable*: A collaborative musical instrument. pages 406–411, 2006.

[5] L. Liu and P. Khooshabeh. Paper or interactive?: a study of prototyping techniques for ubiquitous computing environments. pages 1030–1031, 2003.

[6] J. Patten, H. Ishii, J. Hines, and G. Pangaro. Sensetable: a wireless object tracking platform for tangible user interfaces. pages 253–260, 2001.

[7] J. Patten, B. Recht, and H. Ishii. Audiopad: a tag-based interface for musical performance. pages 1–6, 2002.

[8] J. Patten, B. Recht, and H. Ishii. Interaction techniques for musical performance with tabletop tangible interfaces. page Article No. 27, 2006.

[9] J. Rudd, K. Stern, and S. Isensee. Low vs. high-fidelity prototyping debate. *Interactions*, 3(1):76–85, 1996.

[10] R. Sefelin, M. Tscheligi, and V. Giller. Paper prototyping - what is it good for?: a comparison of paper- and computer-based low-fidelity prototyping. pages 778–779, 2003.

[11] B. Ullmer and H. Ishii. The metadesk: models and prototypes for tangible user interfaces. pages 223–232, 1997.

[12] J. Underkoffler and H. Ishii. Illuminating light: an optical design tool with a luminous-tangible interface. pages 18–23, 1998.

[13] J. Underkoffler and H. Ishii. Illuminating light: a casual optics workbench. pages 15–20, 1999.

[14] J. Underkoffler and H. Ishii. Urp: a luminous-tangible workbench for urban planning and design. pages 386–393, 1999.

[15] M. Weiser. The computer for the twenty-first century. *Scientific American*, 265(3):94–101, 1991.

[16] M. Weiser. Ubiquitous computing. `http://sandbox.parc.com/ubicomp/`, 1996. visited 08.12.2009.

Prototyping for the development of ergonomic interactive surfaces

Maximilian Schenk

Abstract— Prototyping is a prevalent technique to test unfinished systems, forecast later assets and drawbacks or simply visualize various ideas. This work deals with the question how prototyping can be used to adjust ergonomics of a final system as much as possible to human requirements. Therefore it has to be clear which aspects of interactive surfaces influence ergonomics. Five *Ergonomic Input Parameters* are identified: Size, Height, Orientation, Aptitude and the Input Device. Seven recent projects are described to discover how different approaches with different techniques led to a more or less ergonomic adjustment. It will be highlighted which explicit procedures allow the developers to create systems that could be used convenient, rather than systems that are only just working. The early participation of later users at the design process provides a high adaptability of the system which allows the developers to modify their prototype. In combination with multiple prototyping phases the adjustment gets better step by step.

Index Terms—Ergonomics, Interactive Surface, Prototyping, User Centered Design

1 Introduction

"Without investigation of real use, technical feasibility can be meaningless [4]." With these words begins Dan Fittons' work about rapid prototyping. It shows quite plainly how important prototypes can be, at the development of new devices or ways of interaction. In this work the focus will be on the development of ergonomic interactive surfaces. The rise of touch technologies and the ubiquity of computers, displays and interactive devices of all kinds causes the question how to make the next generation of these devices better. One way of improvement is an ergonomic adjustment. Maybe the next generation isn't much faster or bigger or smaller, maybe it's just better to use, like computer displays in the past were characters in some color on a black screen, today we have mostly black characters on a bright screen, because it's easier for us to read.

To give you an overview of prototyping, a short explanation and two examples are provided. After that attention will be put on more ergonomic issues, the parameters that can be adjusted to make a device more ergonomic. What consequences has a determination of a certain parameter on the entire system? Following this I will show how prototypes can or could influence the result using the example of some recent project that used prototyping or not used it. Did the developers put attention on ergonomic aspects, and did they try to find out how to optimize them?

In this work the terms *display*, *table* or *system* are used synonymical. In other context they certainly are not, but here for simplification they all describe an interactive surface.

2 Different Ways of Prototyping

Prototyping in general means to start the real creation of systems in an very early development stadium, instead of thinking trough and then rethink and rethink and then only one time really building it. After building this early prototype the system ist tested in a more or less real situation, to find out what works good and where the weaknesses are and then an optimized version of the system is created, which gets tested again. With this iterative process of creation and feedback the system can be improved in every stage, so that the final system accords to the developers imagination and is build the way it has to be built and not the way the developers think is has to be. In this context the development time and the costs can be reduced [5].

Prototyping can be done on different levels, so there are the two big categories fidelity and functional selection. In table 1 it is shown how Low an High-Fidelity differ from each other in their core aspects.

- *Maximilian Schenk is studying Media Informatics at the University of Munich, Germany, E-mail: Maximilian.Schenk@campus.lmu.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Interactive Surfaces, 2009*

2.1 Low- and High-Fidelity Prototyping

High and Low-Fidelity differ from each other in the following points. Low-Fidelity prototypes are quickly and often cheap constructed, they are supposed to show the concepts and the general idea. They can also sketch different versions, like design variations to evaluate which works best and should be developed furthermore and which should be discarded. Often pen and paper is used to make these sketches, because it's cheap and fast and you need no special skills.

Later I will go into the example of the *Paper Prototyping* technique. The degree of detail is low in relation to the accomplished system but that is no problem, it is volitional because it would not make much sense to cut a picture when you don't know the frame size. The possibilities to interact are limited, but it's enough to see if the project is heading in the right direction or if you better start again with a modified idea. Furthermore lots of different versions can be tested and quickly customized [11].

In contrast to this, High-Fidelity prototypes are in a later level of development they provide real interactivity and simulate the behavior of the system on a higher niveau. The expenses for such a prototype as well as the time to build are higher respectively longer but it is possible to see more clearly how users would interact with or use the system. To pick up the metaphor, if you know the frame size it is reasonable to cut the image to get a better picture of the picture. "Higher fidelity prototypes simulate or animate some but not all features of the intended system [11]."

2.2 Vertical and Horizontal Prototyping

The second big category contains mainly how far the single functionalities are implemented and which functionalities are implemented. A vertical prototype has a very small functional range, but the few functions that are implemented are fully working and not just facile dummys. As the case may be a broad functional range gets cut down to provide in-deep functionality for some key features, which can be tested deeply. This is reasonable to test core functions or if additional functionality is not necessary.

Horizontal prototypes on the other side provide the full rage of functionality on a single level. Everything on this level works as it has to, but there is no functionality in-deep that can be tested. Horizontal prototypes often can be done fast, for example if it is just the interface without a working system underneath. In both versions one part is picked up to be implemented and tested and another part is cut off, because it is unessential for the current evaluation [11]. It can be useful to combine both techniques in different parts of the same prototype [5]. That's the case when, for example the complete interface is clickable but just the central aspect is fully working.

Table 1. Low VS High Fidelity Prototyping [11]

Lo-Fi	**Advantages**
	less time and lower cost
	evaluate multiple design
	useful communication device
	address screen layout issues
Lo-Fi	**Disadvantages**
	limited usefulness for usability tests
	navigational and flow limitations
	facilitator-driven
	poor detailed specification
Hi-Fi	**Advantages**
	partial/complete functionality interactive
	user-driven
	clearly defines navigational scheme
	use for exploration and test
	marketing and sales tool
Hi-Fi	**Disadvantages**
	time-consuming to create
	inefficient for proof-of-concept designs
	managements may think it is real

2.3 Example: Paper Prototyping

The simplest way to explain *Paper Prototyping* is that you make your prototype out of paper. It is a *Low Fidelity Prototype* that can be used in a horizontal way to sketch up different interfaces variations or form factors of a system. (see figure 1) To simulate interactivity, the reaction of the system to a users' input, like the change of displayed content you need someone to play the computer. This person for example has to change the sticky notes, that represent to displays [10]. The more distinct this simulation is, the more blur the bounds to the below explained technique of *Wizard of Oz*.

Nevertheless this simulated version of interactivity works in both directions. When the test user has to type something in, he uses a pen and writes, when a button has to be pressed it gets pressed by touching the paper. The studies of Sefelin et al. [13] compared *Paper Prototyping* with computer based *Low-Fidelity Prototyping* and came to the conclusion that there is minimal difference in the result. The testers mostly preferred the computer based version, but however there are some reasons why P*aper Prototyping* is the treatment of choice.

Advantages of Paper Prototyping [13]

- No restrictions of a framework - You can create whatever you want
- Easy to realize - No special skills are needed
- Fast and cheap

2.4 Example: Wizard of Oz

The need of the *Wizard of Oz* simulation becomes clear when you realize that human communication is very complex and can't be simply done by a computer. Today's technology can simulate some aspects of human communication but often only with a big amount of work. In the development of a system when you decided to use prototyping it is much easier to simulate the simulation instead of creating a simulation system that maybe won't be used furthermore.

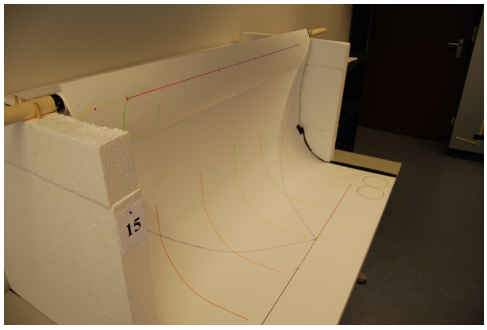


Fig. 1. Prototype N 15 - *Curve* project, made of styrofoam a wooden lath and paper (Image thankfully provided by a *Curve* staff)

This simulation technique is not only used to simulate human communication, most parts of a system for example simple common computer interaction can be simulated too. The idea behind the *Wizard of Oz* is, that a person in the background plays the computer respective the specific part of the system the has to be simulated, because the system or the computer itself has not implemented the specific functionality yet. The tester interacts with the system and does not perceive the wizard in the background [2]. The human wizard can for example react on any input the tester gives, this provides a continuous functionality and the user is not disturbed by elements that maybe are not necessary for the system, but for the correct working simulation of the prototype [6].

3 Ergonomic Input Parameters

Before the question for input parameters could be answered, another question is more important: What is ergonomic? Interactions among humans and other elements of a system can be seen as ergonomic if data and methods are designed "in order to optimize human well-being and overall system performance [8]." The *other elements of a system* are as important as the interactions for this work. This contains many aspects of the prototype which are the *Ergonomic Input Parameters*. Every system has parameters that can be adjusted, but it depends on the system itself if a specific parameter could be seen as ergonomic relevant or not.

An example: If you have a very large public display, the frame width has no influence on how good humans can use the display. When you have a small handheld computer on the other side, the width of the frame is probably essential for a convenient use. In the following there will be described five of those parameters, that could be identified in recent projects. Certainly there are other factors that can be ergonomically adjusted, but I will confine myself to this five as the most important ones for ergonomic prototyping for interactive surfaces.

3.1 Size

The size is one of the most obviously parameters that has to be ergonomically optimized for humans. In this paragraph I will explain why it is so important. Mainly it depends on the future purpose of the system. Is it supposed to be a single workstation or in a conference room to be used for collaborative work? These are basic decisions that influence the raw dimension of the system, but inside this dimension there can be done minimal adjustments that are not less important. So Wigdor et al. [16] found out, that persons preferred the larger field of view, provided by larger devices, even if they have to reach farther for targets at the top and the corners of the screen. Furthermore there can appear privacy issues. When the display is too large content gets displayed large, too. This can be inconvenient for the user, also because he can loose the overview what is displayed. The research of Elliot et al. [3] has resulted, that "large work surfaces should focus on putting primary information near the user and less important information in the periphery [3]".

Nevertheless the *size* as ergonomic input parameter depends on human beings. The form factor of the *Curve* (see 4.1) or the *BendDesk*

(see 4.2) were created on the users biometric characteristics. How large should a digital table be? Data with absolute validity are tough to get, but it has been found out, "that a table measuring 107 cm diagonal is a good minimum size. Observations of smaller table sizes (80 cm diagonal) reveal that users frequently bump elbows and arms with each other while interacting [12]." On a horizontal screen the users tend to lean on their elbows or arms. To avoid false detections of interaction at touchscreen, there have to be touch insensitive areas. This enlarges the entire size of the system [12].

3.2 Height

The second easy to understand input parameter is the systems height. A vertical displays that is mounted too high, can't be interactively used by small adults or children. As a practical example for adjustability of height you could name blackboards at schools. At horizontal interactive surfaces it is a little more difficult, but it also depends on the systems purpose. Wigdor et al. [16] adjusted theirs system in such height, that the user was able to use it while standing or seated. Two sitting heights are mentioned by Ryall et al. [12]: The first is the so called *Lobby Table* it has the height of a coffee-table and can be perfectly used for casual interactions. The second one is a table at *Desk Height*. People can put their legs underneath and work for hours. This height is more suited for productivity tasks.

Table height can also impact reachability and readability of the display for example when because of a wrong height the viewers eyes are too far away from the display [12]. Furthermore the eye height while sitting is a influencing factor, because the head has to be bended when the height is beyond certain constraints [17].

3.3 Orientation

The orientation of the surface means if it's horizontal mounted or vertical mounted. Such a simple variation takes effect on very basic attributes of the interactive surface. Horizontal desktop-like systems are more adapted for physical work like sorting tasks. Users can interact with virtual and physical objects (especially because they don't fall down). Collaborative works is also easier because all four sides of the table can be used [17, 16]. Horizontal table-like displays can be used in two ways. On one side as it is intended for an interactive surface, but on the other side like Wigdor et al. [16] showed, people used it as tables in small meetings and group work. They put things on it and used it as furniture. A pure horizontal workplace has according to Benko et al. [1] negative effects on the users health. They mention that the permanent looking down causes neck strain and that it's not good for the back.

Work environments that are orientated vertically are mostly virtual and without physical objects. Tangibles and other direct touch interaction elements are not necessary because vertical surfaces are better suited for displaying, than directly editing information [17]. The common way of editing content is on a vertical computer-screen, but it is usual to use mouse, keyboard or a stylus and no direct touch input.

Additional to this has the orientation effects on clarity and readability [12]. At a vertical display it's obvious how a photo has to be shown. There is no question: Which side is up? A interactive surface that is mounted horizontal on a table has no way to detect from which side the user is looking at it and maybe shows the photo upside down.

3.4 Aptitude

The parameter aptitude is applicable for horizontal as well as vertical displays [16]. The inclination can advance the ergonomics of a system. The displays Wigdor et al. [16] used were mounted on a sloped surface in order to enable working like on a drafting table. Metrical considerations show that if the top edge of a horizontal display gets raised, the user can easier reach and view the pixels at the top and in the corner of the display. This qualifies the above mentioned dual use of horizontal surfaces. From a certain angle objects would begin to slide across its surface. You should keep that in mind when you adjust the angle.

The neck strain problem, Benko et al. [1] mentioned could be antagonized by a inclination of the surface. The final answer of which aptitude is the best is the following: "Vertical and horizontal interactive displays expose specific assets and drawbacks, and the choice of the appropriate angle depends on the users task [15]."

3.5 Input Device

The last ergonomic input parameter I want to talk about is the input device. The common input devices are mouse and keyboard. They are well established and there is currently no way of not using them in everyday life. It is open to dispute, if a mouse is ergonomically adjusted, but the answer on this question has marginal impact because mice are the common standard input devices in combination with keyboards. Much more interesting are other input devices, like styli, tangibles or simply the users fingers. One might think that fingers are not as precise as a mouse but after a learning phase users can work as precise with their fingers as with a computer mouse [16]. Touch technology can also provide the advantage of multitouch, if the system supports it. Fingers are always availible in contrast to multiple mice, or the computers' support of multiple mice.

A problem with multitouch is, that "some people are hesitant to touch the table simultaneously, especially when they are first introduced to the technology [12]." In a collaborative situation it can happen that people fear to touch another users' hand or bump the arms accidentally with another user. When users know each other, that's not a problem, but when they don't or are even from different cultural areas this can impact the acceptability of the entire system. Unpracticed users could also perform accidental touches with their wrists or cloth, which can't happen with a common mouse. This kind of problems could be avoided by system that cleverly filters erroneous inputs.

Textinput is another important thing and therefor a keyboard works best, "bare fingers are insufficient for text input [12]". Tangibles can be ergonomically adjusted, but a lot of things can get wrong even with the basic form. It is hardly possible to give a general statement how they can be adjusted. Wimmer et al. use for their *Curve* a "combination of WIMP and touch technology [17]". Maybe that's the most ergonomic adjusted model of input devices. People who are inexperienced with touch technology get along due to usage of the classic combination of keyboard and mouse, which is also used for things like text typing. There is still the option to interact by touch which opens new ways of interface design.

4 Ergonomic Prototyping or not?

In the following part I will describe and analyze recent research projects that developed and/or observed interactive systems. I want to highlight if the researchers in these projects applied prototyping and if they had an advantage of doing so. The focus will be if the scientists put attention on ergonomic aspects. Did they avoid mistakes, or was the outcome the same, as if they would not have used prototyping? On the other hand I want to show projects in which mistakes could have been avoided with the right prototyping. Furthermore I will disclose how and which way the prototyping was used or could have been used.

4.1 Curve

The first project I want to mention is the *Curve* [17] project. Within this project the researches wanted to close the gap between physical and virtual desktops. As seen above, horizontal surfaces are more appropriate for some tasks than vertical surfaces are and vice versa. The *Curve* is an approach to combine these two characteristics in one single desktop-sized interactive surface. The basic form is "a soft curve forming one large, L-shaped surface [17]." At the thought of ergonomics they put a lot of work in the finding of the right dimensions. They decided to gather the values form the dimension parameter in three steps. The basic measures where taken from literature [9] to adjust the curve to a average german. After this additional constraints have been discovered in discussions with designers and cognitive psychologists. The undetermined factors have been defined by a study with 18 different prototypes made of paper. These prototypes represented all possible combinations of the three open factors. The first factor was the top edge of the vertical surface in relation to eye height of the user. The higher the edge is the larger is the effective seen display. They picked

two values: The average users' eye height +5cm and -5cm. The two other open factors were the inclination of the vertical surface and the radius for the curve. Values used in the study have been 5, 10, and 15 degrees for the surface inclination and 5, 10, 15 cm for the curve radius. Based on these values they build 18 different prototypes with a fixed size of 120 cm. The students, taking part in the study had to fill out questionnaires and must test every single prototype and rank them. After evaluation of the study the researches could name the preferred constellation. The *Curve* desktop they built has the values shown in *figure 2 a)*.

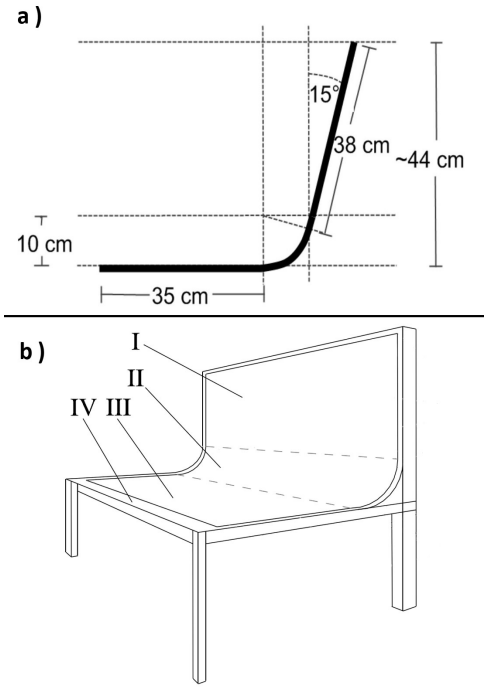


Fig. 2. a) *Curve* measures seen from the right [17], b) *BendDesk* and it's ergonomic parts of the display (**I** vertical screen (39" x 19"), **II** curve to connect I and III (radius: 4"), **III** horizontal screen (39" x 15"), **IV** non-interactive border (39" x 3")) [15]

At this project they put attention on good preparation work and aspects which could not be prepared have been prototyped in a relative fast an inexpensive way. Due to the building of 18 versions of paper they could figure out the best combination of values. The testing of these different configurations made them get a most ergonomic adjusted final version.

4.2 BendDesk

The *Benddesk* [15] project is very similar to the *Curve* project. The researchers at this project also tried to combine the established output technology of vertical displays with horizontal displays, which are more efficient for example drawing tasks. They also perform this combination by connection the two displays by a curve which allows dragging from one displays to another. In the absence of detailed information there is not much about to say how they derived the dimensions of the *BendDesk*. The dimensions of the displays are, except the width, similar to the ones of the *Curve*. One could suspect that there was an analogue basic ergonomic consideration. The curve radius is nearly the same as at the *Curve* (4" are about 10.16 cm). The "preliminary user tests [15]" seem to led to the knowledge that users prefer a greater curve radius when they slide up with the finger. Sadly they did not present why and how they came to the conclusion that a radius of 4" works best. The finding of the other parameter values, would have been interesting, as well. Especially when the focus is on the drag-from-horizontal-to-vertical-display feature, the not existing inclination of the vertical displays could be questioned. The *Curve* prototypes have pointed that a backwards inclination of the vertical display is convenient for the user. This ergonomic aspect perhaps was disregarded at the *BendDesk*. Perhaps it would have attracted attention with an adequate way of prototyping.

An ergonomic aspect the *BendDesk* researches considered is the additional non-interactive border. This is one of the factors that hardly comes up in a prototype made of paper. A high fidelity prototype maybe could avoid this and show disadvantages of a design other ways of prototyping can't show. That's probably the reason why the *Curve* researchers did not add a non-interactive border.

Concluding the results of both projects are similar, with little differences, but the elaborate prototyping of the *Curve* researchers led to more ergonomically adjustment in some fields.

4.3 Interactive group learning

The next project is an interactive group learning system for children in elementary school [14]. The pupils had to construct a physical model of a town in a playful way *(see figure 3 a))*and then get the environmental effects of their activity presented on a virtual screen *(see figure 3 b))*, to "verify knowledge they have acquired from a schoolteacher or a textbook [14]". Like in other projects the systems' requirements were asserted at first in interviews with specialists. In this case the specialists were the teachers and the pupils. It came up that image-processing technology with a camera is not suitable for a classroom, because pupils are not able to set up and dismantle such equipment. The technology of choice was sensor-based detection.

Before building the final system, they made some different sized paperboard prototypes with various numbers of grids on it. A clear idea of the hardware is necessary, because it's harder to change after a certain point than software. These prototypes were without detection, one of the researchers simulated the systems input manually. At this test, in which 30 pupils took part, they used relative simple prototypes made of paperboards and a little bit of software. The size of the boards had to be more than 60 x 60 cm, so that four to six pupils can work at one system, but not too large, because of their shorter arms pupils lean over the board to reach fields far away from them. Another crucial point is the size of the game pieces, "because pupils would have difficulty manipulating pieces that were too big or too small [14]". This is a nice example of ergonmic adjustment of tangibles. Furthermore children get along better with user-interfaces with characters, that with graphical representations, a point that only can be found out at when testing with kids.

After all the final system was developed based on the requirements which became clear after testing prototypes in classrooms. Lots of possible failures haven't been made because of early prototyping and good preparation work. After establishing the final system it came up that the boards where slightly too heavy for one pupil to carry. This depends on the used sensing hardware. It would be hard to name this a failure, but it shows quite fine how important it is to understand the prototyping process as a circle where parameters influence each other and have to be considered in every phase of developing. Sugimoto et al. [14] observed, that requirement given from small children are not a 100 percent correct, because they can't express themselves in way grown up can do and probably are not aware of what the requirements really are.

4.4 Interruptibility Prediction

The focus of the following project is not on the development of an ergonomic system in a physical meaning but it is prototyping on the communication level. It's a feasibility study on the basis of the prototyping technology *Wizard of Oz* [7]. Humans need only one moment to cognize if another human can be interrupted at the moment or if it's appropriate to not disturb. Computers don't have this ability what in daily life leads to computers that either always interrupt instantly and present what ever they have to present, or never interrupt and wait for the user the request the content. This feasibility study wants to figure out, if there would be appropriate sensors a computer based system would be able to predict if it is suitable to interrupt at a specific moment.

a) b)

Fig. 3. a) The *Interactive group learning* system used in a classroom with elementary school pupils [14], b) An example of visualizing a town being constructed on the board [14]

As initial point of the study they made a long-time video recording with audio, of four subjects in a similar single-office environment. The subjects were asked periodically to rate their current interruptibility. Based on these recording the researchers identified 23 events or situations that could give information about the subjects interruptibility. They used some machine learning techniques to develop predictive statistical models, which should be able to predict a humans' interruptibility. The problem on testing the models is that, at present, there are no appropriate sensors to detect the parameters. To deal with this they used the *Wizard of Oz* technique and simulated the sensor input manually - but in a real situation. So they could test the effectiveness of their models and prove the feasibility of human interruptibility prediction, if there would be suitable sensor technology.

4.5 Hermes 1

The *Hermes 1* [4] is a system installed at office doors. Users can leave messages and this allows asynchronous communication between them, like sticky notes could do. The mounting outside an office door aims to combine private an public elements. The basic decision that "without user studies, technical feasibility can be meaningless [4]" led to a split of the development into phases. This phase-based cycle in combination with rapid prototyping and a deployment of the systems in a very early state, provides a maximized ability to adjust and improve the system.

The system contained only a small range of functions, this low complexity makes it very easy for the testers to use the *Hermes 1* messaging system and ensures a high reliability for the whole system. At the beginning they aimed to develop the core functionality with only two *Hermes 1* systems given to friendly users. (*See figure 4 - a)*) A weakness is that friendly users' adjudgement could be affected from their friendliness.

While the complete development process the researchers logged the entire usage of the system to get detailed feedback where the weaknesses are and where the systems runs as it should. In every prototyping cycle the amount of installed devices was continuously increased, in the same way the study participants were extended. From only friendly users at the beginning to ones that are not so familiar with this kind of technology. The splitting of development process into phases provides another advantage, the focus on which characteristic has to be enhanced, can be changed for the reason that a goal-orientated development is possible. Due to the deployment in a very small extent at the beginning they could figure out a "major issue impacting reliability.[4]" A great deployment maybe would have make it difficult to discover that people standing in front of the *Hermes 1* block the wireless signal and disable the working of the system.

In the next phase with more devices this issue was resolved. In later phases, based on feedback of study participants the amount of interaction possibilities was increased. Extending the functional range at the beginning would not have been reasonable, because the requirements just became clear after evaluating the feedback from the users.

The focus in this project was not on ergonomics, but this project shows how a system can be success when user feedback is included in all stages of the developing process. If the requirements are verbalized from the ones who use the system, the system gets adjusted to them.

4.6 SPAM

Another system in whose development process *Hermes 1* displays have been used is *SPAM*, that stands for "SMS Public Asynchronous Messenger [4]." The system aims to expand the possibilities of communication in a working environment beyond the existing ways to communicate, like telephone, fax and email. The idea behind is to enable the sending of SMS not only to persons mobile phones but also to a public screen. Public in this case does not mean the market square but rather an office with an well-arrange amount of people. So it is possible to send a message to a place and not to a person.

To not overstrain the user or frighten him off, the developers attached great importance to reliability and easiness of usage. Before starting the construction the researchers did two weighty things to discover how the system could be adjusted to later users and usage. The first was to gather ethnographic data on usage. Due to this move is easy to make fast to realize and associated with only low financial charges it gives serviceable insights into users requirements. Certainly this was only the first step, because all aspects could hardly be covered by unspecific analysis.

The other important preliminary work was to set up a Participatory Design Workshop. Under usage of props, including the above mentioned *Hermes 1*, the later users took share in the development process. In this workshop they rapid prototyped different scenarios to discover which works best with the test users. This way later users are able to give useful feedback in a very early state of development, for example they could refuse a designers' idea before he puts effort on an idea that is probably less promising. The cheap the obtaining of ethnographic data was, the expensive is it to make such a workshop.

Bringing the findings of both processes together the requirements for the later *SPAM* system became clear. Fitton et al. [4] used "off-the-shelf" software and hardware components to quickly build a version that could be deployed and tested outside the lab in a real environment. During these tests every action of the participants were logged to evaluate them later. This way the researchers were able to get a very fast and pure feedback of the actual use of the *SPAM* system. Based on this feedback they made some modifications very quickly.

In this project to the focus wasn't on ergonomic topics in a straight physical way, as well. The size for example plays a subordinate role meanwhile the adjustment to the users requirements and their participation in the development process were essential. In later project phases they had not much to modify which led to the conclusion that good preliminary work can counteract a large amount of prototyping cycles.

4.7 Hermes 2

The success of the *Hermes 1* displays brought the developers to use it as starting point for another project, with the focus on how the "physical placement and design [4]" of wireless connected displays influence their usage in a community. Based on the *Hermes 1* they started the development of the *Hermes 2* system [4]. As mentioned at the *Hermes 1* project less effort was put on ergonomic inputfactors as defined in chapter *3*. The *Hermes 2* should be more ergonomically adjusted, which has to be reached by a suitable development process. Parameters should be configured in a way users are comfortable with. Some of the parameters - they sadly don't name all of them - are display size, number of devices or user interface layout. In the first phase of the development the researchers focused on two key aspects: The physical from-factor, which means for example the size or the thickness and the general display configuration users can handle most suitable. Therefore they created six different styled prototypes.

To get a situation that is mostly realistic they mounted displays in an environment that's similar to the later actual usage. Here the ergonomic inputfactors height, aptitude and orientation attract interest, even if the researchers are not aware of this fact. The participants of

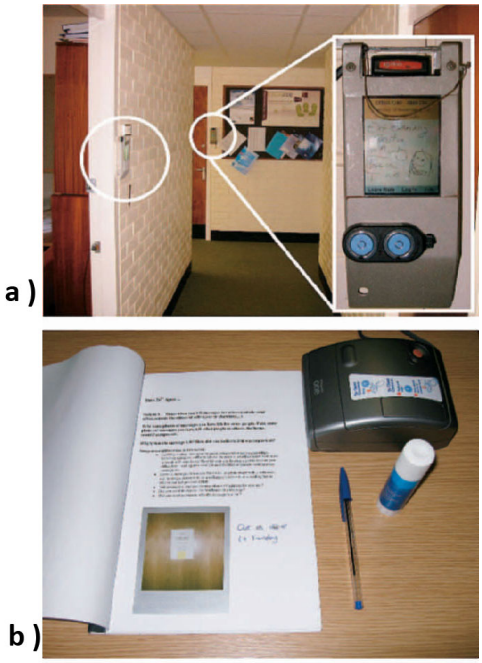


Fig. 4. a) Hermes 1 deployment [4], b) a Hermes 2 probe pack, containing a diary, a camera, glue and a pen [4]

the study, the persons that will get a display, were filmed when the developers showed them a "tour around the showcases [4]". Through the guidance when introducing the showcases the users could be brought to give high quality feedback. They combined different aspects of various prototypes and told the researches precisely which configuration they would prefer. Such a large scaled activity is associated with a lot of time and money for the building of the prototypes and nevertheless with organizational effort.

All participants of the showcase demonstration were equipped with a probe pack. "these packs contain a diary, instant camera, pen, and glue [4]" (*see figure 4 - b)*). This probe pack should help the later users to journalize their messaging behavior. Every day they had to note down messages they had left for themselves or others and messages others had left for them. To have a better imagination of the surroundings of the message photos could be taken with the camera and glued into the diary. This analog recoding lasted for seven days.

The early consultation with the end users and their involvement in the development process, in order they were able to give high quality feedback and discuss with the developers generated a mostly ergonomically adjusted final system. The paper prototyping for the user interfaces only had limited success, but Fitton et al. [4] admit that this was caused by a not extensive enough realization.

5 Conclusion

What have all these projects shown? Is prototyping reasonable when the final system has to be ergonomically adjusted? In the following I want to excerpt answers from the just inspected projects. The *Curve* project is distinguished by good preparation work and a great amount of built prototypes to evaluate the final parameter configuration. The researchers at the *BendDesk* project led to similar results without a large-scaled prototyping. The *Curve* developers explicit focused on ergonomics and gathered the slightly better result which can be traced back to the extensive prototyping. Testing prototypes with the later users early in the development process and continuous recheck of the requirements were the recipe for success of the *Interactive group learning* project. Here it comes up that a frequent verification of the requirements can be useless if the source of the requirements is unreliable. At the feasibility study for *Interruptibility Prediction* the prototyping worked as it has to. Here the focus was on the evaluation a system before it is finished. Phase based development was the idea behind the *Hermes 1*. In every development cycle the prototypes have been enhanced respectively adjusted to the users requirements. The later users, in this case they were reliable, have been included in the system design process and gave precious feedback. Due to this a high flexibility and a maximized ability to adjust were provided. In *SPAM* project the end user have been included as well and their preliminary work and certainly their experience on this sector did not require many prototypes. *Hermes 2* was developed with early end users feedback as well and supported by prototypes to visualize potential systems.

The question at the beginning of this work was if prototyping is reasonable. The answer to this question is ambiguous. Prototyping is reasonable if the requirements are not a 100 percent clear. A good preliminary work, on the other side, can reduce the necessity of prototyping. Picking up the point *ergonomics*, it can be very useful to prototype and include the later users in very early levels of development to get fast feedback. Understanding prototyping cyclic allows to adjust the system in every loop more and more to the end user.

References

[1] H. Benko, M. R. Morris, A. B. Brush, and A. D. Wilson. Insights on interactive tabletops: A survey of researchers and developers. 2009.

[2] N. Dahlbäck, A. Jönsson, and L. Ahrenberg. Wizard of oz studies: why and how. In *IUI*, pages 193–200, 1993.

[3] A. Elliott and M. A. Hearst. How large should a digital desk be?: qualitative results of a comparative study. In *CHI '00: CHI '00 extended abstracts on Human factors in computing systems*, pages 165–166, New York, NY, USA, 2000. ACM.

[4] D. Fitton, K. Cheverst, C. Kray, A. Dix, M. Rouncefield, and G. Saslis-Lagoudakis. Rapid prototyping and user-centered design of interactive display-based systems. *IEEE Pervasive Computing*, 4(4):58–66, 2005.

[5] C. Floyd. A systematic look at prototyping. In R. Budde, K. Kuhlenkamp, L. Mathiassen, and H. Züllighoven, editors, *Approaches to prototyping*, pages 1–18, Berlin, 1984. Proceedings of the Working Conference on Prototyping, Springer.

[6] J. Höysniemi, P. Hämäläinen, and L. Turkki. Wizard of oz prototyping of computer vision based action games for children. In *IDC '04: Proceedings of the 2004 conference on Interaction design and children*, pages 27–34, New York, NY, USA, 2004. ACM.

[7] S. E. Hudson, J. Fogarty, C. G. Atkeson, D. Avrahami, J. Forlizzi, S. B. Kiesler, J. C. Lee, and J. Yang. Predicting human interruptibility with sensors: a wizard of oz feasibility study. In G. Cockton and P. Korhonen, editors, *CHI*, pages 257–264. ACM, 2003.

[8] International-Ergonomics-Association. What is ergonomics? Website, 2000. Available online at `http://iea.cc/browse.php?contID=what_is_ergonomics`; visited on December 8th 2009.

[9] W. Lange and A. Windel. Kleine ergonomische datensammlung. TÜV Media, 2008.

[10] L. Liu and P. Khooshabeh. Paper or interactive?: a study of prototyping techniques for ubiquitous computing environments. In G. Cockton and P. Korhonen, editors, *CHI Extended Abstracts*, pages 1030–1031. ACM, 2003.

[11] J. Preece. Prototyping, 2005. Available online at `http://hamilton.bell.ac.uk/btech/hci/hcinotes17.pdf`; visited on December 8th 2009.

[12] K. Ryall, C. Forlines, C. Shen, M. R. Morris, and K. Everitt. Experiences with and observations of direct-touch tabletops. In *Tabletop*, pages 89–96. IEEE Computer Society, 2006.

[13] R. Sefelin, M. Tscheligi, and V. Giller. Paper prototyping - what is it good for?: a comparison of paper- and computer-based low-fidelity prototyping. In G. Cockton and P. Korhonen, editors, *CHI Extended Abstracts*, pages 778–779. ACM, 2003.

[14] M. Sugimoto, F. Kusunoki, and H. Hashizume. Design of an interactive system for group learning support. In *Symposium on Designing Interactive Systems*, pages 50–55, 2002.

[15] M. Weiss, S. Voelker, and J. Borchers. Benddesk: Seamless integration of horizontal and vertical multi-touch surfaces in desk environments. 2009.

[16] D. Wigdor, G. Penn, K. Ryall, A. Esenther, and C. Shen. Living with a tabletop: Analysis and observations of long term office use of a multi-touch table. In *Tabletop*, pages 60–67. IEEE Computer Society, 2007.

[17] R. Wimmer, F. Schulz, F. Hennecke, S. Boring, and H. Hußmann. Curve: Blending horizontal and vertical interactive surfaces. 2009.

Prototyping for the Development of Ergonomic Interactive Surfaces

Eduard Vodicka

Abstract— Interactive surfaces like tabletops or walls are currently a popular topic in research. They offer a new way of interacting with a computer, making a contrast to traditional desktop environments. Interactive surfaces can allow multi-touch and multi-user interaction and thus create new working scenarios. To be most suitable to the user, ergonomic aspects should also be taken in mind. This paper deals with the prototyping process when developing such ergonomic interactive surfaces. Prototyping is necessary to evaluate concepts and find errors and problems, so it should always be part of the design process.
At the beginning of this paper an overview of different prototyping techniques like paper or hardware prototyping is presented. The concept and design challenges of ergonomic interactive surfaces are discussed in the next step to understand the requirements in the design process. On the basis of exemplary projects the actual way from an idea to the final prototype is shown. It occurs that hardware prototypes are the usual choice when evaluating a project. Paper prototypes instead are used rarely but should be the first choice, especially when designing ergonomic devices.

Index Terms—prototyping, tabletop, ergonomics, interaction, gestures

1 Introduction

The term interactive surface generates a contrast to the common desktop computer paradigm. With interactive surfaces it is possible to explore other forms of displaying information and interacting with it. The desktop metaphor usually presents the data on a monitor, the user can interact with the system by using a keyboard and a pointing device like a mouse or a touchpad. Interactive surfaces allow direct interaction with the device or the place where the information is presented for one or more users. It is not determined what exactly an interactive surface consists of. It could be a simple touchscreen hanging on a wall or a flatscreen embedded in a tabletop or a projection on a wall or any other combination of technologies. These devices have in common that the users can operate with them directly, for example with their hands using gestures *(see figure 1)*.

While designing such surfaces, several problems can occur. An idea has to be realized in some way, which means technical realization in the first place. It is often difficult to build a new device with existing technology. Sensors for example could not be as precise as needed for a correct detection of the user's inputs. It is also important to think about the interaction techniques that have to be accepted by the users. Another point that should be taken into consideration is ergonomics. When working with ergonomic rules, the chances that the system can be operated by a heterogeneous group of people are increased. The users will accept the project and ideas faster, when it is easy to use for them.

A way to solve such problems during the designing process or simply to check and validate ones ideas is evaluation. By letting users test the product and ideas, problems can be identified and new solutions can be found. It is recommended to begin these evaluations early and to repeat them during the development process. A prototype can be used to present the users something they can work with. So the basic approach when designing a new interactive surface seems to be to build a prototype that matches the ideas, evaluate it and continue the development with the new insights. Of course this scenario is not only suitable for interactive surfaces but for all kinds of hardware and software developments.

This paper gives an overview about the different possibilities to realize these prototypes and show the goals that can be achieved by using these different kinds of prototyping. Another aspect of this paper is the design of interactive surfaces and what it means to call them ergonomic. Finally, those two aspects are combined to show how prototyping is actually used in the process of designing new interactive surfaces. This way it can be analyzed, which prototyping techniques are suitable and which are not. In the conclusion, a recommended way of prototyping can be shown.

Fig. 1. A multi-user scenario on a tabletop display [26].

2 Prototyping

Building prototypes is an essential part in the process of designing new interfaces in software or hardware. It is a fast and cheap alternative to build something based on ones ideas to test the producibility or to present it to others. These others could be customers who want a presentation of the made progresses or the users of the future product. Letting the users interact with the prototypes can give valuable insight in the usability of the ideas, which is an important part of the evaluation. The designers can manifest their ideas with prototypes and thus bring their hypothesis to life to test them [18]. By these evaluations it is not only possible to prove the concepts, but also to learn more about the problems that should be solved [12]. In this way prototyping is an important step before building the final product. To have enough time to analyze and implement the insights brought by the evaluations, the first stages of prototyping should begin early in the design process.

Prototyping allows to filter unimportant aspects like the colors of an interface or the shape of the buttons. A prototype is never complete, which is its strength [18]. The designer does not have to think about

- *Eduard Vodicka is studying Media Informatics at the University of Munich, Germany, E-mail: eduard.vodicka@campus.lmu.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009/2010*

how exactly the product will be build and which material or technology it will require. Instead he is free to focus on the concepts on his mind.

Basically there are two different approaches to build a prototype. The low-fidelity (lo-fi) and the high-fidelity (hi-fi) prototype. Low-fidelity means a simple non-functional sketch of the ideas to present certain aspects of the products such as the user interface. Low-fidelity prototyping is often associated with paper prototypes because this is an often used technique [25]. In contrast high fidelity prototypes seem to be much more advanced. That could be for example an already working piece of software or hardware or a simulation that already looks like the final product. These two approaches both have advantages and disadvantages. Lo-fi can be build very quickly and focus on certain problems that should be evaluated.Hi-fi can be used for a detailed proof-of-concept or to sell an idea to others. But hi-fi can also cause problems when focusing the work on a bagatelle and therefore forgetting the core issue [25].

There is also another categorization for prototypes, distinguishing between horizontal and vertical prototyping. The vertical version implements only selected functions of the final product but these functions are fully operational and ready for use. The horizontal version in contrast tries to give an overview over as much of the planned functions as possible, but they are not implemented in detail and so not fully usable [8].

To make building of prototypes easy, developers can use tools and frameworks that support them in the design process. These tools allow to sketch prototypes with predefined components and can therefore speed the work up [12].

2.1 Paper Prototypes

As mentioned above, a common possibility to implement a lo-fi prototype is to draw ones ideas on paper. Sketches of the user interface of an application or a website can be made this way without writing a single line of code *(see figure 2)*. There has to be no worrying about colors, icons or images, instead the developers can focus on layout and interaction [29]. A paper prototype is easy to build, it is cheap and fast to create. All that is needed are some drawings of screens and menus that are composed together. After having finished the prototype it can be used not only to check the visualized ideas but also for evaluation. The users can interact with the 'application' by touching it. An operator acts as a computer and changes the screens. Using a paper prototype the testing is independent from technology and implementation issues and changes can be easily adopted. Problems with the interface can be found and considered before having finished the application.

Instead of drawing the screens on paper to build a low-fidelity prototype it is possible to make them on a computer. The user can interact with such a mockup of an application by using a display and input devices like mouse and keyboard. These mockups can be implemented interactively or by using the Wizard of Oz approach *(see section 2.4)*. In comparison to paper prototypes the visual component and the look-and-feel are more important in computer based prototypes.

Studies have shown, that at early design stages the gathered information using paper or computer based prototypes is quite similar [27] and that people do not respond negatively to the incomplete and dirty looking paper prototype [29]. So the criterias of choosing a prototyping technique can be for example time, because it is much faster to build a paper prototype than a computer based or even interactive one [19]. As discussed in by Liu et al. [19] a paper prototype does not have to be suitable at later points in the design process. Concrete technical ideas and more complex interaction scenarios can require a more complex type of prototype.

2.2 Prototyping in Hardware

When developing a new software application, it seems to be easy to build an appropriate prototype. The development of new hardware devices can be more complicated in this point of view because at some point a working piece of technology has to be assembled. At the beginning of the process a simple sketch of how the device should look like

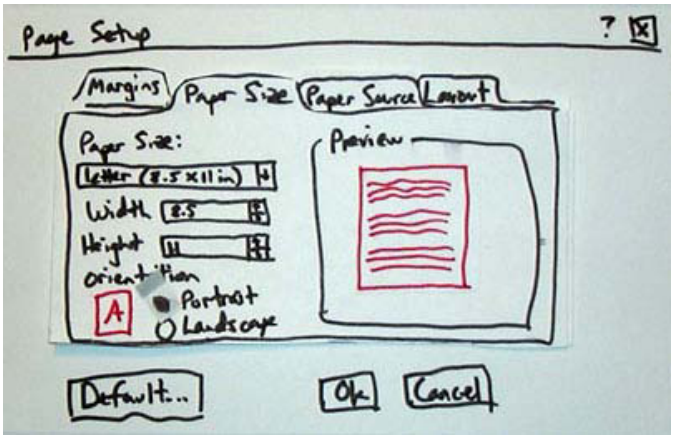


Fig. 2. Paper prototype of a graphical user interface [29].

can be drawn [3] or a mockup using materials like paper or styrofoam can be build [33]. Such mockups can be used for user studies.

To achieve real functionality, a working piece of hardware is needed. But at this point building the finished device could be too early. Conducting studies on the finished product is ineffective because it will be expensive to change aspects that caused problems. What is needed is an easy to build hardware prototype. There are toolkits that allow to do such a thing, hardware components that consist of microprocessors, different kinds of sensors or input and output interfaces. These components can be connected to one another and to a computer *(see figure 3)*. The developer does not have to worry about the communication between these components, he can use a software framework to program them and instruct them what to do. Hartman et al. presented an example for such a toolkit [12]. Using hardware toolkits is a way to build interactive hardware prototypes without developing new technology. It is a goal to make that process just as easy as sketching in software [13].

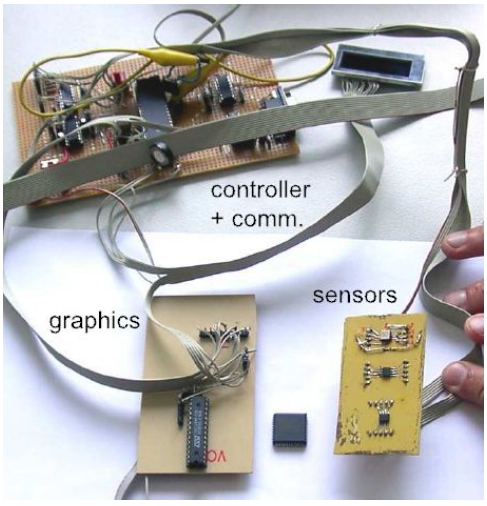


Fig. 3. Hardware prototype build with a toolkit [3].

2.3 Video Prototypes

Traditional prototyping techniques have their limitations when something is actually build. It is possible that the resulting prototypes are not portable or not really usable. Most of all they are incomplete and they possibly do not show the actual idea of the project to someone who is not directly involved, for example the customer. Prototypes are designed to test specific functions or interfaces and thus do not have the functionality of the final product. A good way to show someone

the idea and the vision behind a product and to present the planned functions without implementing everything is to produce a video that shows how the product will look like and the ways it can be used. A video breaches both software and hardware limitations and so it is possible to write a storyline where people interact with a device or an application that does not exist yet. In that way plans and ideas can be presented to an audience that possibly does not know anything about the product.

Because of the possibilities offered by producing a video, there are some important points that should be taken in mind, as discussed by Bruce Tognazzini [30]. It is easy to build a mockup of a complex piece of hardware that is very hard to produce in reality, so it should be assured that the real product is not exaggerated. Of course in a film the actors do exactly what the script says and so every interface looks easy to use. A video does not free from actually designing and testing the interface and thinking about the implementation of the shown interaction techniques. Having a script also means that that the users in the video should not operate the device without problems all the time. It is important to create situations that feel real. By considering these points it is possible to make a realistic and convincing video prototype that carries the spirit of the ideas.

2.4 Wizard of Oz

Another approach in prototyping is to use humans to simulate computers. These prototypes have to be looked at from different points of view. The user's point of view is a fully functional device or software application that he can begin to use instantly. From the developer's point of view, the prototype is only a mockup which is operated by people who react on the user's interactions to maintain the illusion of functionality. The user is not aware of what happens between his input and the following reaction of the used device.

Most people behave differently when communicating with a human or a machine. Because of this, Wizard of Oz studies can give insights on how the users will interact with a new system. Other forms of prototyping that are directly operated by a human like paper prototyping cannot achieve this [14, 4]. The Wizard of Oz approach is suitable for situations which require user input that is hard to interpret for a computer and thus hard to implement, such as voice recognition [9], complicated textual input or gestures [14]. In such situations a wizard can be used at early design stages when a complex system is not yet available. The wizard can monitor the users and adapt to them and thus gain information about their behavior. In order to act like a machine, for example react quickly and avoid typing errors, simulation environments that provide templates can be used.

Dahlbäck et al. [4] conducted a study that analyzed whether the users are aware of the fact that they are actually communicating with a human. It shows that in general the users do not notice the system is simulated (assuming the prototyping setup has no design errors). This leads to an ethical question because the study participants are deceived about the true nature of their conversation partner. It should be avoided to bring the users in embarrassing situations.

3 Interactive Surfaces

Displays have become bigger during the last decades. From classic small computer displays they have evolved to huge flatscreens and tabletop displays and even wall-sized screens. During this time, the form of interaction has evolved, too. Beginning with touchscreens, the idea leads to interactive surfaces, meaning huge displays with which the users can interact directly. An interactive surface is not just a display anymore, it is not only an output device, the possibility of direct touching makes it an input device at the same time [28].

Interactive tabletops are a very popular example for interactive surfaces. The idea is moving away from a single-user setup with everybody having his own vertical display to a more collaborative scenario *(see figure 1)*. Sitting face to face at a table shall support this scenario. In fact, researchers have the opinion that collaboration or multi-user environments are one the advantages of interactive tabletops [1]. To allow cooperative working the tabletops must support multi-touch or multi-gesture input which occurs to be another advantage. Collaborative applications can be for example games, photo browsing applications [26] or brainstorming sessions [1]. Classic single-user scenarios such as text processing or web and email applications are not very popular on tabletops yet due to the lack of efficient text input, vague pointing and ergonomic issues when working alone at a large horizontal display [1].

The new usage scenarios, interaction designs and of course the appearance make interactive surfaces look different from the known desktop computers. People often do not realize that the same technology that is operated by common Windows or Unix based systems is working underneath. They consider interactive tabletops to be something completely new and not only another form of input/output devices for already used computers [26].

Fig. 4. A single user using a tabletop to work with traditional everyday software [32].

3.1 Design Challenges

Building new interactive surfaces can lead to different problems that have to be solved in the design process. There is little common knowledge of interaction possibilities and user behavior when facing an interactive surface. The technical realization has to be considered, too.

3.1.1 Technical Aspects

The first challenge is the question how to get the surface to become a display. The different approaches like front or rear projection or using a flatscreen have different advantages and disadvantages. Hiding a projector under a table means there is no room for feet left but projection from the top leads to occlusion. According to Ryall et al. [26] this is not a great concern because the shadowed area is not larger than the area hidden by the users hand anyway. Flatscreens can lead to resolution problems when users are located too close to it. A display on a wall often simply does not allow back projection because there is no room in the back. This means that a projector has to be installed in front of the wall. Since interactive surfaces do not have to be flat but also can have different curved forms, another problem occurs. It is needed to create a seamless high resolution image on these surfaces. Because a projector can only create a flat image, multiple projectors whose projections overlap can be a solution. The equations to calculate the correct transfer and projection parameters or the intensity blending which is needed to achieve such a seamless projection are introduced by Raskar et al. [24].

Desktop computers and even classical touchscreens allow only one pointing device. So when realizing multi-touch or multi-gesture user interfaces on interactive surfaces a new technical solution must be found. The common solution here is to use optical systems to detect objects that touch a screen or are located close to it. It is possible to

use infra-red sensible cameras or optical sensors that can be build into an existing Flatscreen and thus making it multi-touch capable [15]. Recognizing multiple input points does not yet allow full multi-user support. By only sensing touches it is not clear whether the system is operated by one ore more users or which pointer belongs to who. The Diamondtouch system [6] introduces a way to clearly distinguish between different users. Electrical signals make it possible that each user has its unique ID when interacting with the tabletop *(see section 4.2)*.

People sitting around a tabletop display do not have equal access to the displayed information. Text documents or images have an orientation that allows only the users sitting on the right side to see them in a proper way. The other participants of a collaborative work have to either look on the objects from a wrong direction or change their orientation manually. This is not considered to be a problem with small documents because humans are capable of recognizing text even in a wrong orientation [26], still there are approaches to solve this problem technically [21, 16]. A setup of multiple projectors and mirrors can give each user his unique view and thus rotating the objects on the screen so that they are easily visible.

3.1.2 Interaction

Large interactive displays like tabletops or walls are not suitable for the old interaction metaphor using mouse and keyboard. New interaction techniques have to be developed to ensure these new devices can be used effectively and properly. One possibility is the direct interaction with the surface using fingers or devices like pens to touch it. Gesture or pressure based interaction can be realized this way. Techniques that allow moving objects on a large screen or between different devices have been compared by Nacenta et al. [22]. It is important to differentiate between the possibilities that the target is within or out of hands reach. A far target does not allow to simply move it with a movement of the hand. In this case gestures can be used to direct an object to a distant place. Another approach is to use a miniature map of the whole environment to reach distant places without actually moving there physically. It should be considered that a finger is not as precise as a mouse cursor and that the interfaces have to be designed in a way to support this fact [26]. Interaction with hands and fingers can be supported by tangible devices that may have additional control elements, for example buttons that invoke a menu [11].

Direct interaction is not possible in all situations, especially when the used surfaces are large, for example an interactive wall. In these situations remote pointing devices can be used. A laser pointer [5] for example makes it possible that a user can interact with a wall-sized display without standing directly in front of it and reach all areas, even the distant ones.

3.1.3 User Behavior

Though an interactive surface and its user interface can be thought-out in every detail, it is possible that the users will not accept it or have no need for it. It can be useful to observe people using an interactive surface in a real world scenario without clear instructions what to do with it or how to operate it. There are two studies on that topic, one conducted by Ryall et al. [26] focuses on multi-user support. The other one, conducted by Widgor et al. [32] observes a single user *(see figure 4)*. These studies point out the problems when using everyday software applications such as office systems, web browsers or email clients. Inaccurate pointing input by using fingers and the lack of an efficient way to type text are the main causes of this. It is interesting that users begin to operate the tabletop systems with only one finger although they are multi-touch capable. After being informed of this fact, they begin to experiment with gestures. This may change in the future because of the rising presence of multi-touch devices such as the iPhone in everyday life. Having enough training, the users are able to adapt and begin to use both hands to operate the systems more efficiently. But not only multi-touch scenarios are difficult at the beginning, multi-user approaches experience the same problems. People hesitate touching the tabletop at the same time as others, being afraid of accidentally interfering with the work of others or even touching them. The problem of coming in contact with another person is even more important when people in the group do not know each other. In this situations it is even more difficult to conduct corporate tasks. A well-rehearsed team can use the advantages of a multi-user environment much faster.

Another detail that has to be taken into consideration when designing interactive tabletops is that they are actually used as tables. People put cups and other objects on it and lean their arms on it while working. This can lead to problems when these touches are interpreted as input. An insensitive area on the edges is needed or a technology that can distinguish real touches and large objects just standing on the surface. Hygienic issues are also important, because many people hesitate to touch something others have touched before and thus do not want to use their fingers and hands to interact with the surface.

3.2 Becoming Ergonomic

A developer who wants to design interactive surfaces that will be accepted by the users has to think about ergonomic aspects. Ergonomically formed interfaces should be the most suitable for the users, so that they like to work with them and, which is also very important, do not suffer any physical problems when using them over a long period of time. Part of the ergonomic thoughts is the size of the device and the alignment of the user interface. The measures and the inclination of a tabletop can determine whether all areas are comfortably reachable or lie out of hands reach so the user has to stand up or move to interact with them. Thoughts on a tabletops size and height have been made in different studies [7, 26]. These variables can depend on the usage scenario of the interactive surface. A single user may need a smaller and angled desk to comfortably reach all edges, a group on the other hand needs a larger table where the people have enough room to seat and enough personal space on the display so they can work with it. A table that is used for casual interaction such as games or photo browsing can be at coffee-table height, a place where people actually work for a longer period of time should have desk height. Depending on the scenario decisions about these characteristics can be made early by conducting a user study to find out what people are comfortable with [33]. Not only the device itself has ergonomic aspects, the means of control are an important part, too. For interactive surfaces that are controlled by direct touch, the designers have to implement a set of gestures. These gestures should not only be easy to learn and intuitive but also ergonomic and thus not physically stressing. To achieve this goal it is necessary to think about biomechanics and to find out which hand or finger movements are easy to perform and which are difficult [23].

Lightning characteristics are another part of the ergonomics [2]. To easy recognize the content of a display, its brightness should not be too high or too low. Contrast, sharpness and reflections on the surface have to be considered, too. These variables do not only have an instant effect (dark and blurred screens are hard to read), working with poor adjusted devices can tire the eyes and even lead to permanent sicknesses.

Ergonomic aspects should be considered and tested before actually building or at least finishing an interactive surface. As a first step it can be helpful to look into a compilation of ergonomic data [17] that can contain information such as average human height, arm length or eye height. Considering these values can give insight if an idea will work or not.

4 Using Prototypes

As seen in the last section, interactive surfaces are still a quite new research area. There are no standard approaches that can be followed and many things have to be developed from scratch. User needs have to be identified and ergonomic aspects have to be taken in mind. This means conducting user studies. The research projects have different approaches and purposes and thus may require different kinds of prototyping. Still in most cases building a prototype is necessary. Some of these projects on interactive surfaces are presented in this chapter.

4.1 Size of a Digital Desk

Setting the dimensions of an interactive tabletop affects the ergonomic aspects of the device. Ame Elliot and Marti Hast wanted to determine these dimension by comparing different devices in a user study [7]. They used a desktop computer with mouse and keyboard input, a tablet pc and a tabletop with a stylus. The participants had to fulfill a sketching task and a sorting task with images. The used devices were not self build, existing commercially available hardware with fixed dimension was used. The results suggest that for sketching tasks, tablet or tabletop are preferred and for sorting the desktop computer is the right choice. This study basically compared three different output devices and two different input devices. Since the size of the desk was fixed and only the angle was adjustable it is not possible to say what size a tabletop should have or if the results would be different when using a smaller or bigger version.

4.2 DiamondTouch

The DiamondTouch system [6] introduces a multi-touch and multi-user capable technology for interactive surfaces. The touches are not recognized through an optical system but an array of antennas build into the surface. Each antenna emits a unique signal which is passed through the body of the user who is touching it into a receiver in the chair or in the floor. This way it is possible to determine what antenna a user has touched and thus where on the screen he points and which user has touched the antenna. The DiamondTouch is able to assign each recognized touch point to a specific user. This enables new possibilities in collaborative work. Because the surface is insensitive to pressure and optical effects and needs a closed circuit to recognize a touch, objects can be put on it without affecting the interaction.

The authors of the DiamondTouch suggest that their technology can be used in different sizes and resolutions. It was not part of their research to determine which size and setup was the most suitable for interactive surfaces, they only introduced a new technology that can be used for different devices. Because of this they did not conduct a user study, they only build a small hardware prototype *(see figure 5a)* to proof that the technology is working.

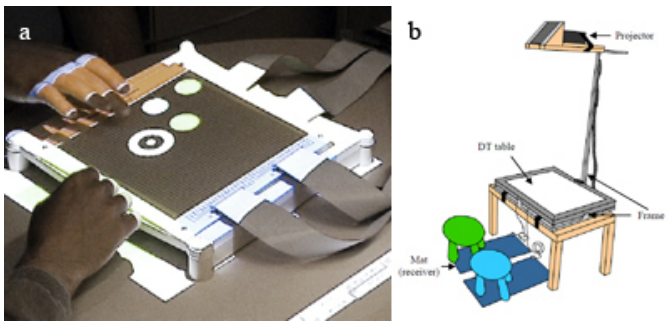


Fig. 5. Prototype of the DiamondTouch technology (a) [6] and a sketch of the Fantasy Table (b) [20].

4.3 Fantasy Table

The DiamondTouch technology has been used for different studies on interactive surfaces [26, 32]. The Fantasy Table project [20] also relies on this system. This project wants to analyze the possibility to use interactive surfaces to support fantasy games of young children in the age from three to four years, in order to introduce them in the world of technology. The decision to use an interactive tabletop *(see figure 5b)* and its dimensions were proceeded by not specified prototyping and user studies. The final design was implemented using existing components, the DiamondTouch as hardware and Adobe Flash as Software. The first study revealed major interaction problems, most of them as result of the software implementation (objects that should be moved are too small) but also problems with the hardware itself. The children tended to move while using the tabletop and so lost contact with the receiver mat on the floor which terminated the interaction.

The software issues were solved by changing the game setup. The objects became larger and the geometrical complexity of the displayed environment was reduced. A second study using the new prototype was more successful. The children actually began to play with the game on the tabletop instead of just playing with the possible interaction techniques.

4.4 cueTable

Another project dealing with interactive tabletops is the cueTable [10]. The goal is not to use new technology or to improve the interaction possibilities but to research the use of such devices in collaborative scenarios. The cueTable prototype was a self made multi-touch capable device. A pong like game with teams consisting of two people that play against each other was used to study the users behavior in cooperative and competitive situations. The measures of the prototype were not derived from a preliminary user study because ergonomic issues were not considered. When playing the game, some users held their hands in exhausting positions or had problems with the interaction techniques. Another problem was the delay between touch and feedback due to technical latency of the system.

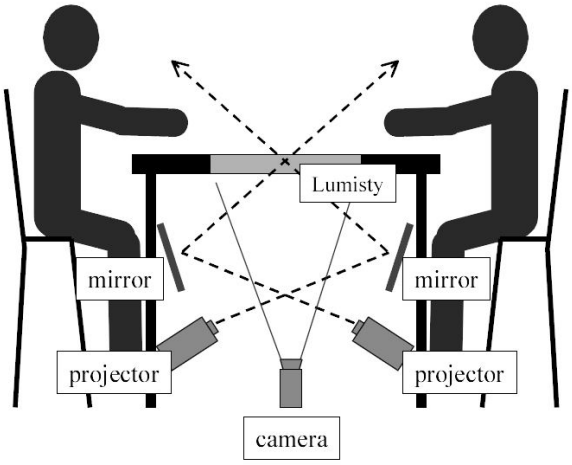


Fig. 6. Schematic setup of the Lumisight table [21].

4.5 Lumisight Table

As mentioned above, the orientation of documents and objects on a tabletop surface might be a problem in collaborative work. These effects occur when multiple users are working with a single view from different positions. The Lumisight table [16, 21] offers a possible solution by offering more than one view. A different image is projected to each side of the table with the objects on screen being rotated so that every user can see the content from the correct perspective. The systems works with multiple projectors and mirrors *(see figure 6)* and a screen material that becomes transparent or opaque depending on the viewing direction. The position of the objects is not altered, so collaborative interaction and discussion is still possible which would not be the case if each view showed objects at another position instead just rotating them.

Ergonomic aspects were not considered when building the prototype. The hardware does not allow to put feet under the table, the dimensions were set without evaluation. The user study that was made focused only on the question whether the developed system increases the performance of the people who work with it, which was the goal of the authors.

4.6 Curved Surfaces

There are currently multiple projects on combining horizontal and vertical interactive surfaces. The Curve [33] and the BendDesk [31] project both work on creating a device that blends both types of work

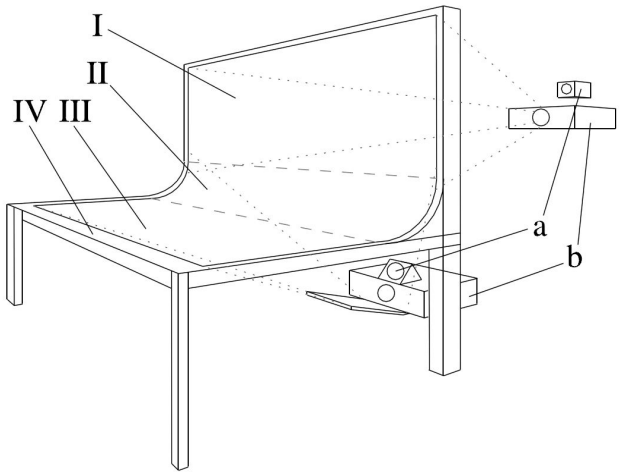


Fig. 7. Hardware setup of curved tabletop blending horizontal and vertical surfaces [31].

spaces. Vertical and horizontal displays or surfaces are suitable for distinct kinds of work and are not always interchangeable. Using both kinds in different devices to combine the advantages has been practiced so far, the idea of the current projects is to create a seamless interactive surface that has a horizontal and a vertical part. Basically it is a tabletop that fades into wall through a small curve *(see figure 7)*. This construction allows it to use both parts of the device equally as it is most adequate to the task and to position documents or objects on the screen as needed. For example, the vertical part can be used as a display like in classic desktop environments, the horizontal part as an input possibility for drawing tasks.

Both projects are working on a hardware prototype and possible software applications. The dimensions of the BendDesk has been determined after conducting a not specified user study. The Curve project began with paper prototyping. Different setups were created and evaluated resulting in the dimensions like height and angle of the vertical part or width of the device and which are most suitable to the users thus trying to reach the best ergonomic properties as possible.

5 Discussion

In chapter 4 some projects on interactive surfaces covering different approaches and having different goals were presented. All these projects used the concepts of prototyping in some form to achieve their goals and to validate their results. The technique that was used in all cases is hardware prototyping that means that all projects had a functional device in the end or are currently building it. It seems suitable to build such a prototype at some point, if it is to determine whether an idea is technically realizable or to conduct a user study to evaluate the interaction techniques. There exist possibilities with different coverage and complexity, you can build a prototype from scratch and thus destine its appearance and dimensions or use existing technology and extend it. The choose depends on the research goals, a novel interactive surface will supposably have to rely on new and self build hardware. A project that primarily researches interaction techniques or forms of user cooperation can use an already working device and just design the proper software.

It is noticeable that all regarded projects actually build working devices, meaning hardware and software. No Wizard of Oz approaches were used in the user studies. The cause may be that this technique may work well for textual or speech input or generally input forms that give the operator some time to react. Surfaces that are based on touch and gesture interaction require an instant response on unpredictable user movements which would be difficult to handle for a wizard, but not impossible [14] *(see section 2.4)*. It would be a problem in user studies when the system would give false reactions on the users' input. They could recognize that it is operated by a human or think that it is not working properly and thus not suitable for everyday use.

Just one of the projects [33] used paper prototyping or at least did not mention it. Using this technique, the design and the measures of a planned device could be tested and experimented with without spending too much time developing in a wrong direction. Instead this phase is skipped most of the time, the hardware prototype is build directly with no further tests.

Like paper prototyping, ergonomic issues are mostly left out when developing a new technology or an interactive surface. Of course they do not have to be considered in projects that have different purposes but should be when designing a new device or studying the behavior of the users. Thinking about the dimensions or the form of a surface and evaluating them leads to a more usable device and avoids problems when testing the final hardware prototype. Just setting those variables can mean that the final product will be too small or too large or have areas that are not reachable for normal grown persons. Such mistakes can cost time and money to fix them and can cause that the users simply do not like the product because it is exhausting to work with it.

After having discussed the common prototyping techniques and projects on interactive surfaces, it can be said what workflow seems recommendable. When trying to build an ergonomic interactive surface it is important to check the ergonomic issues first, before building the actual product. Paper prototyping seems the most suitable for this task. A mockup *(see figure 8)* or different variations can be build without much effort and then evaluated to find out the best design. The next reasonable step then would be the technical realization and implementation of the needed software. This leads to a hardware prototype. This can be used for new user studies that deal with interaction on the surface. The results can be used for further research. Of course a concept video for presentations can be made during the whole process.

Fig. 8. User study to determine the dimensions of the Curve using a paper prototype [33].

6 Future Work

For future research it would be helpful to move the focus more on ergonomic issues. They are a vital part when designing hardware and interaction techniques that should be actually used by people in their everyday life. An interactive surface that brings physical stress to its users and is not applicable for a longer period of time has failed in HCI. This situation should be brought to the researchers minds. Another point is the effect of prototyping techniques on the development of ergonomic interactive surfaces. The right choice of prototyping can save time and money because errors can be detected earlier. A detailed study on how the choice of prototypes affects the development and success of the project would be helpful, this could result in a guideline of the correct workflow when designing an ergonomic device.

7 Conclusion

There are different types of prototyping serving different purposes. When looking at the development of interactive surfaces it is common to build a hardware prototype at a final stage of the design process. Other forms of prototyping like the Wizard of Oz approach are difficult to adapt to the scenario of interactive surfaces. Paper prototyping is rarely used but that does not mean they could not be useful. Especially when taking in mind ergonomic aspects, building a simple mockup can be useful to wipe out design errors. Ergonomics are mostly leaved out of the design considerations, even if it would be important for the project and could help to avoid mistakes.

References

[1] H. Benko, M. R. Morris, A. B. Brush, and A. D. Wilson. Insights on interactive tabletops: A survey of researchers and developers. Technical report, Microsoft Research, Redmond, WA, USA, March 2009.

[2] U. Bräuninger and E. Grandjean. Lighting characteristics of visual display terminals from an ergonomic point of view. In *CHI '83: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 274–276, New York, NY, USA, 1983. ACM.

[3] A. Butz, M. H. Gross, and A. Krüger. Tuister: a tangible ui for hierarchical structures. In J. Vanderdonckt, N. J. Nunes, and C. Rich, editors, *IUI*, pages 223–225. ACM, 2004.

[4] N. Dahlbäck, A. Jönsson, and L. Ahrenberg. Wizard of oz studies: why and how. In *IUI '93: Proceedings of the 1st international conference on Intelligent user interfaces*, pages 193–200, New York, NY, USA, 1993. ACM.

[5] J. Davis and X. Chen. Lumipoint: Multi-user laser-based interaction on large tiled displays. *Displays*, 23(5):205–211, 2002.

[6] P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM.

[7] A. Elliott and M. A. Hearst. How large should a digital desk be?: qualitative results of a comparative study. In *CHI '00: CHI '00 extended abstracts on Human factors in computing systems*, pages 165–166, New York, NY, USA, 2000. ACM.

[8] C. Floyd. A systematic look at prototyping. In R. Budde, K. Kuhlenkamp, L. Mathiassen, and H. Züllighoven, editors, *Approaches to prototyping*, pages 1–18, Berlin, 1984. Proceedings of the Working Conference on Prototyping, Springer.

[9] J. D. Gould, J. Conti, and T. Hovanyecz. Composing letters with a simulated listening typewriter. In *Proceedings of the 1982 conference on Human factors in computing systems*, pages 367–370, New York, NY, USA, 1982. ACM.

[10] T. Gross, M. Fetter, and S. Liebsch. The cuetable: cooperative and competitive multi-touch interaction on a tabletop. In *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, pages 3465–3470, New York, NY, USA, 2008. ACM.

[11] F. Guimbretière, M. Stone, and T. Winograd. Fluid interaction with high-resolution wall-size displays. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 21–30, New York, NY, USA, 2001. ACM.

[12] B. Hartmann, S. R. Klemmer, M. Bernstein, L. Abdulla, B. Burr, A. Robinson-Mosher, and J. Gee. Reflective physical prototyping through integrated design, test, and analysis. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 299–308, New York, NY, USA, 2006. ACM.

[13] L. E. Holmquist. Sketching in hardware. *interactions*, 13(1):47–60, 2006.

[14] J. Höysniemi, P. Hämäläinen, and L. Turkki. Wizard of oz prototyping of computer vision based action games for children. In *IDC '04: Proceedings of the 2004 conference on Interaction design and children*, pages 27–34, New York, NY, USA, 2004. ACM.

[15] S. Izadi, S. Hodges, A. Butler, A. Rrustemi, and B. Buxton. Thinsight: integrated optical multi-touch sensing through thin form-factor displays. In *EDT '07: Proceedings of the 2007 workshop on Emerging displays technologies*, page 6, New York, NY, USA, 2007. ACM.

[16] Y. Kakehi, M. Iida, T. Naemura, Y. Shirai, M. Matsushita, and T. Ohguro. Lumisight table: An interactive view-dependent tabletop display. *IEEE Computer Graphics and Applications*, 25(1):48–53, 2005.

[17] W. Lange and A. Windel. *Kleine ergonomische Datensammlung*. TÜV Media, 2009.

[18] Y.-K. Lim, E. Stolterman, and J. Tenenberg. The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas. *ACM Trans. Comput.-Hum. Interact.*, 15(2):1–27, 2008.

[19] L. Liu and P. Khooshabeh. Paper or interactive?: a study of prototyping techniques for ubiquitous computing environments. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 1030–1031, New York, NY, USA, 2003. ACM.

[20] E. I. Mansor, A. De Angeli, and O. de Bruijn. The fantasy table. In *IDC '09: Proceedings of the 8th International Conference on Interaction Design and Children*, pages 70–79, New York, NY, USA, 2009. ACM.

[21] M. Matsushita, M. Iida, T. Ohguro, Y. Shirai, Y. Kakehi, and T. Naemura. Lumisight table: a face-to-face collaboration support system that optimizes direction of projected information to each stakeholder. In J. D. Herbsleb and G. M. Olson, editors, *CSCW*, pages 274–283. ACM, 2004.

[22] M. A. Nacenta, D. Aliakseyeu, S. Subramanian, and C. Gutwin. A comparison of techniques for multi-display reaching. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 371–380, New York, NY, USA, 2005. ACM.

[23] M. Nielsen, M. Strring, T. B. Moeslund, and E. Granum. A procedure for developing intuitive and ergonomic gesture interfaces for hci. In A. Camurri and G. Volpe, editors, *Gesture Workshop*, volume 2915 of *Lecture Notes in Computer Science*, pages 409–420. Springer, 2003.

[24] T. W. Ramesh Raskar, Jeroen van Baar. Quadric transfer for immersive curved display. Technical Report 2004-034, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, January 2004.

[25] M. Rettig. Prototyping for tiny fingers. *Commun. ACM*, 37(4):21–27, 1994.

[26] K. Ryall, C. Forlines, C. Shen, M. R. Morris, and K. Everitt. Experiences with and observations of direct-touch tabletops. *Horizontal Interactive Human-Computer Systems, International Workshop on*, 0:89–96, 2006.

[27] R. Sefelin, M. Tscheligi, and V. Giller. Paper prototyping - what is it good for?: a comparison of paper- and computer-based low-fidelity prototyping. In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pages 778–779, New York, NY, USA, 2003. ACM.

[28] C. Shen, K. Ryall, C. Forlines, A. Esenther, F. D. Vernier, K. Everitt, M. Wu, D. Wigdor, M. R. Morris, M. Hancock, and E. Tse. Informing the design of direct-touch tabletops. *IEEE Computer Graphics and Applications*, 26(5):36–46, 2006.

[29] C. Snyder. Paper prototyping. `http://www.cim.mcgill.ca/~jer/courses/hci/ref/snyder.pdf`, 2003. visited 19.11.2009.

[30] B. Tognazzini. The "starfire" video prototype project: a case history. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 99–105, New York, NY, USA, 1994. ACM.

[31] M. Weiss, S. Voelker, and J. Borchers. Benddesk: Seamless integration of horizontal and vertical multi-touch surfaces in desk environments. In *Extended Abstracts of Tabletop '09*, 2009.

[32] D. Wigdor, G. Penn, K. Ryall, A. Esenther, and C. Shen. Living with a tabletop: Analysis and observations of long term office use of a multi-touch table. In *Tabletop*, pages 60–67. IEEE Computer Society, 2007.

[33] R. Wimmer, F. Schulz, F. Hennecke, S. Boring, and H. Hußmann. Curve: Blending horizontal and vertical interactive surfaces. In *Adjunct Proceedings of the 4th IEEE Workshop on Tabletops and Interactive Surfaces (IEEE Tabletop 2009)*, Nov. 2009.

Prototyping in Physical Computing - Sketching in Hardware

Robert Kowalski

Abstract— Sketching in hardware provides an opportunity to quickly and repeatedly present and evaluate product ideas during their development. Within this field, hardware prototyping toolkits provide a flexible and feasible approach to quickly develop a rough "hands-on" experience for potential users or customers and collect their feedback. This paper addresses this way of prototyping in the following way: First of all, the broad field of potential toolkit users will be structured into four unique user profiles, which represent different expectations and knowledge levels. These profiles are based upon a toolkit advisor framework, which will be introduced in this paper. Nevertheless, since there are more criterions, besides the two the framework is based on, the second part will cover further aspects on how to differentiate the current available toolkits. With these basics in mind, the following paragraph will discuss a selection of toolkits, ranging from scientific ones, which are not publicly available, to versions, targeting for style and exclusivity. This is followed by two more "exotic" examples, which focus on paper computing and mobile prototyping. With this foundation of user profiles, criterions and examples, a discussion will evaluate the question which toolkit is suitable for whom.

Index Terms—Prototyping, sketching, hardware, tangible, interaction, toolkit, mock-up

1 Introduction

Prototyping, physical computing and sketching in hardware are the three major terms in this topic. For a better understanding, a short introduction into these concepts will be given and as an example for their practical relevance, the connection towards commercial product development will be provided.

According to Porter's value chain [25], product development is very important and should be a company's core competence. Besides many research and engineering tasks, prototyping is a driving factor within this process, which describes the creation of product prototypes.

Prototype derives from the Greek words "protos", meaning "first" and "typos" for "impression"[1], which already reveals the basic meaning: Giving a potential user or customer a first impression of a possible tool or product and offer a possibility for the designers to test their design hypotheses [5]. Besides visualization, prototypes enable companies to continuously evaluate the results during development, which may prevent them from "throwing a product over the wall", resulting in unusable products, for example. Therefore, prototyping is an iterative process, allowing the designers to repeatedly improve quality over time [24]. During these iterations, the "product" evolves from rough low-fidelity mock-ups to detailed high-fidelity solutions, which are very close to the final result. As already implied, low-fidelity prototypes stand for early ideas and try outs in order to find a customer-focused approach. Since these early sketches are fast and easy to produce, many iterations towards the right solutions can be done in short time. As soon as the features and the interaction design are fixed, high-fidelity prototyping comes into play. In contrast, this is not about fundamental features or ideas, but it addresses details like the visual design of buttons for example.

On first sight, physical computing may not be directly linked with prototyping. However, it becomes essential the minute it is not about software products anymore, but physical or hardware products that can actually be touched and are able to read and compute data from the "physical world". An exemplary cycle would be the data input via sensors, followed by the analysis of the collected information, possible resulting in an output or feedback through actuators. Sensors may include pressure-, magnetic-, photo- or tilt sensors, which gather specific data or are able to recognize certain situations. If such a situation is triggered, a microcontroller takes over, processes the signal and forwards another signal to an output device, like a Liquid Crystal Display or servomotor [12]. Taking this into account, it is obvious, that new ways of prototyping are needed in this domain, in order to be able to quickly draft a system, that specifically reacts to certain factors in the environment by combining sensors, one or more microprocessors and output devices.

This process is called "Sketching in Hardware". In general, this basically describes the activity of creating running hardware prototypes, to be able to actually show it to customers or users for evaluation. Therefore, sketching "is an essential part of any design process" and whereas for example a graphic designer sketches on paper, a designer of an interactive prototype does not only focus on the static look, but on the "dynamic properties", like the "essence of a proposed system" [14].

Taking this into account, physical computing is a powerful tool for the designer to convey an idea. Yet, this is also the crux of sketching in hardware. Since average designers tend to be not much experienced with electrical engineering and software programming, new tools are needed to lower the entry barrier for designers. Fortunately, prototyping toolkits are available, which not only provide easy access, but furthermore heavily support the iterative process by providing reusable parts and microprocessors, making prototyping much cheaper and less time consuming, as well.

As you might imagine, there are countless toolkits in development and use. In order to provide a common entry point to this topic, this paper will start with two short motivational examples, to show off the power of toolkits. After that, the different types of users will be identified, which is a first approach towards toolkit classification. Since there still remain a large number of prototyping platforms, criterions will be introduced, that allow further subdivision. Having these basics in mind, this paper will address current toolkits, as well as examples from the upcoming fields of paper- as well as mobile prototyping. Finally, a discussion will determine, which toolkit is suitable for whom.

2 Motivational Examples

Before the introduction of the user profile framework and the enumeration of exemplary toolkits, two projects will be provided, in order to show the strengths and possibilities of hardware prototyping toolkits.

The first example has been taken from a sketching with hardware block course at the University of Munich and puts the word "rapid" into rapid prototyping. In other words, this example demonstrates, that it is possible to quickly deliver excellent prototypes. The result of two days work by two electrical inexperienced students can be seen in figure 1a. The concept was to have illuminated cubes, which change their color according to the cube's side, which faces up. Besides the recognition of orientation, the cubes were also able to detect if another

- *Robert Kowalski is studying Media Informatics and Technology Management at the University of Munich and the Technical University of Munich, Germany, E-mail: robert.kowalski@campus.lmu.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009/2010*

[1] Online Etymology Dictionary - http://www.etymonline.com

cube has been stacked upon, put below or has been pushed side by side. In all these cases, the cubes not only changed their colors, but even took on the same ones.

The construction of a laser harp[2], represents the impressive possibilities prototyping toolkits offer. A laser harp, as seen in figure 1b, can be best described as a regular harp, which has laser beams instead of strings. When a beam is interrupted, a MIDI event is generated and send to a synthesizer, which creates sounds depending on the MIDI event's properties[3].

These were just two prototyping examples, but they already provide a glimpse at the hardware prototyping toolkits' potentials.

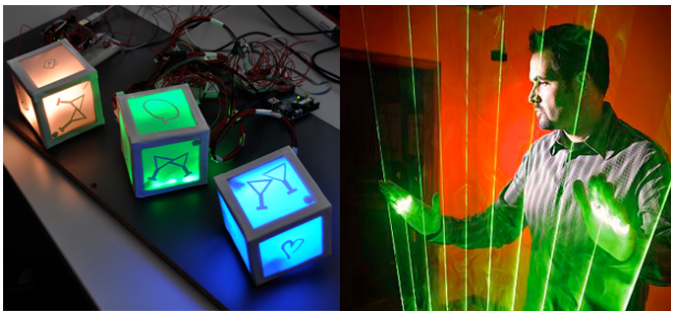


Fig. 1. a) Prototype of the "What Do I Want Cube" b) a laser harp by Stephen Hobley; Sources: own photo, photo from `http://www.stephenhobley.com/build/`

3 User Profiles

When approaching the topic for the first time an average user faces two problems: Firstly, the sheer amount of available toolkits and the lack of information how "powerful" they are. This addresses a typical problem to find a fitting toolkit, which delivers all required functions to "get the job done". The second problem deals with the height of the entry barrier, respectively novelty and confirmation according to Weizsäcker [29]. If it is too high, the toolkit might be suited best for the task, but unless the user is not able to work with it, the project will fail anyway. In order to solve these two problems of information overload and lack of information, a framework, inspired by the Boston Consulting Group Portfolio Matrix [28] and the 2D mapping of various toolkits by Camille Moussette [22], has been developed. The goal of this framework is, to enable users to quickly determine which toolkit is suitable for them, depending on their level of knowledge and the desired complexity.

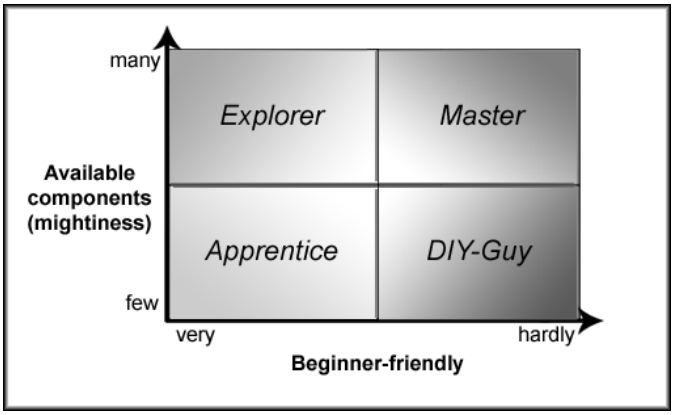


Fig. 2. Toolkit advisor framework; Source: own illustration

[2]http://hacknmod.com/hack/create-techno-with-a-laser-harp/
[3]http://www.youtube.com/watch?v=sLVXmsbVwUs

This toolkit advisor, displayed in figure 2, is based on a two dimensional graph, whose y-axis corresponds to the amount of available toolkit components, which is directly linked to the toolkit's mightiness. Additionally, the graph's x-axis reflects the beginner-friendliness. The definitions of these two criterions will be provided within the according subsections below. By subdividing the graph into four categories, a two by two matrix is created, which holds the different user profiles: Apprentice, Explorer, Master and DIY-Guy, which will be described below.

3.1 Apprentice

The Apprentice is a newcomer in prototyping and hardware sketching, but is eager to learn. This person has little or no knowledge, when it comes to electrical engineering, soldering or programming and is looking for an easy introduction into the topic. Preferably, these early steps should deal with electronics as little as possible and programming is done via visual programming or in high level languages.

3.2 Explorer

After gaining some experience, the Apprentice reaches the limits of the beginner toolkits. The major limiting factor is the lack of additional components for more complex projects. These would require the adoption of other toolkits, which facilitate the use of a broader range of input and output modules. By doing so, the Apprentice becomes an Explorer and now works with toolkits, which are more versatile, but remain to be beginner-friendly. Nevertheless, visual programming is substituted by high level languages.

3.3 Master

As soon as the user got acquainted with complex toolkits and high level programming, she or he becomes a Master. As a result, low-level programming languages are no limiting factors concerning toolkit selection anymore. Furthermore, the user is no longer dependent on pre-assembled parts, like boards with soldered microcontrollers.

3.4 "DIY-Guy"

The Do-It-Yourself Guy represents the approach of an almost complete abandonment of pre-assembled parts. From the board's circuitry, over the microcontroller programming via assembler, to even the construction of own sensors, this user does nearly everything by himself. However, this profile is not very common within commercial hardware sketching, since the process of iterative prototyping should be as fast and cheap as possible.

4 Criterions for Differentiation and Filtering

Alongside the two criterions amount of available toolkit components and beginner-friendliness, additional differentiation factors are possible and might be even necessary, in order to further break down the amount of toolkits in consideration. Therefore, the following criterions pose as an extension to the advisor framework, which can be put in use in order to further filter the toolkits within a user profile. Moreover, these factors have been selected, because they represent limiting project properties. For example, within the user profile "Explorer", there are a certain number of toolkits for selection. If there are no project team members, who are capable of programming C, it would be advisable to filter out these toolkits, which rely on that language and choose from the remaining sketching platforms. In order to integrate further criterions into the matrix, it is possible to alter the radius of the circle, which represents the toolkit. For example, the mightier a toolkit is, the "thicker" becomes the circle.

4.1 Available Components

This factor is an important one, since it is used in the advisor framework. This is due to the fact, that the amount of available components is not only directly linked to the mightiness of the toolkit, but also to the possible complexity.

Considering the components, they can be subdivided into three categories: Input, output and data processing/computing modules. Input modules typically are all kind of physical sensors like pressure,

tilt, light, etc. In contrast, actuators, displays or LEDs are often used for output. When it comes to the computing modules, there are also a wide variety of different microcontrollers, differing in architecture, speed and price for example.

A further important issue, which greatly contributes to the amount of available components, is the question if any modules from electronic stores can be used or just specific ones that are bundled with the toolkit. Therefore, this should be considered as well, when the factor's value is determined on the axis .

4.2 Mightiness

As mentioned above, the toolkit's mightiness is a criterion, resulting directly from the amount of available components. This is due to the fact, that with the amount of available modules, also the number of unique module combination rises, which ultimately leads to an increasing number of possible task solutions. The more solutions there are, the mightier the toolkit becomes.

4.3 Beginner-friendly

This criteria is used in the framework as well, since this is an important issue for the target group of designers, for example. Basically it takes into account how much previous knowledge is needed, in order to work productively with a toolkit. The driving sub factors are programming and electrical engineering skills. Concerning programming skills, the scale is based on the language's suitability for beginners as well as its popularity. Popularity has been taken into account, because it is an indicator on how much support a beginner can find in the many software developer communities on the web. In order to "measure" the popularity, the Tiobe Programming Index[4] has been selected, since its rating mirrors this aspect. On the hardware side, this factor takes into account, how easy it is to assemble the modules. Less beginner-friendly toolkits for example, would urge the user towards soldering works, instead of giving him the possibility to just stick the components together via magnets or into a breadboard.

4.4 Location of power supply

The location of power supply is by far no trivial question in later stages of the prototyping process. The two common options are either the "onboard" integration or the external supply via an USB port. The important thing to keep in mind is the concept of the future product. If it should be a wireless or freely movable object, then a wired approach might not be the best idea, considering the design in general or the quality of the user feedback from the studies. Furthermore, if the power supply should be integrated, then the size of the battery must be taken into account as well.

4.5 Location of processing power

This criterion is very similar to the above one. But instead of power supply, this is about the location of the processing power, meaning the microcontroller. The two location options remain the same as well, including the implicit problems that might occur when using it: On the one hand there is the onboard version and on the other hand there is the possibility that the processing power is provided by an additional computer. But there is a third option, which might come up in the following years and is closely connected with the current trend towards cloud computing. In this case, the data processing would be done on servers inside the internet.

4.6 Programming language support

As mentioned within the introduction of this section, the programming language should not be disregarded. Moreover, the programming language supported by the toolkit plays an important role as well. Naturally, this criterion is closely linked to the beginner-friendliness and therefore more focuses on the languages themselves. Consequently, this is a simple, but easy to apply filter.

[4]http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

4.7 Platform-independency

The importance of this criterion rises with the growing popularity of Windows platform alternatives, like Mac OS X or Linux. The cliché is, that "creative people", like designers, often work on Macs, whereas highly technical oriented people rather work on Linux. Although this is no reliable data, there is undoubtedly some truth in it, which makes the platform-independency an important criteria for the users and especially teams, who work on heterogeneous systems.

5 Examples of common toolkits

The following paragraph will examine some examples of toolkits, which are currently in use. Since there are countless platforms available, this paper focuses on giving examples for each of the different user profiles of the advisor framework. For every toolkit, general information on the development and on the functionality will be given. Furthermore, the above introduced differentiation criterions will be covered as well.

Fig. 3. a) I-CubeX toolkit b) Lego Mindstorm NXT 2.0; Sources: [1], [21]

5.1 Lego Mindstorm

In a nutshell, Lego Mindstorm is the extension of the Lego bricks experience towards physical computing. In particular, it is the continous use of the bricks, which makes this toolkit special for everyone, who ever played with Lego.

The current Version NXT 2.0 of Lego Mindstorm has been released in August 2009 and can be seen in figure 3b. Like the previous versions, NXT and the Robotics Invention System (RIS) are based on a robotic kit, which has been developed by the MIT. Here, the well known Lego blocks serve as a basic structure, on which the several modules get attached to. These input and output devices are wired to the NXT, a central programmable unit, which is equipped with an ARM 32bit microcontroller.

This toolkit offers several components, that can be used with it. It includes color, touch, light, sound, ultrasonic, compass and an accelerometer sensor for input. For output purposes the user has the choice between several servo motors and furthermore the NXT itself has a speaker and a display. Therefore the amount of available components is limited and consequently the mightiness is not high.

In order to program the NXT, the visual programming language LabVIEW from National Instruments has been integrated into the software development kit ROBOLAB from Lego, which can be used on a PC or Mac. After the program is finished, it can be uploaded via USB or Bluetooth onto the central unit. This combination of a visual programming approach and the usage of the well-known Lego bricks and design for the modules, leads to a high beginner-friendliness.

As already mentioned, the processing power is situated inside the NXT unit as well as the power supply. Another option is the use of a rechargeable battery pack.

When it comes to programming languages, the Mindstorm toolkit is highly supported from third-party languages. From Visual Basic to

C and C++ over Java and Python, there are many languages, that have been taken as a basis for the development of own NXT programming languages, like brickOS[5],Ch[6], Lego.NET[7], leJOS[8] or NXT-Python[9].

A further advantage of this support is, that through the many different languages all users of the computer platforms like Windows, Mac OS X or Linux can prototype with this toolkit [4], [15].

5.2 I-CubeX

In contrast to the other toolkits, I-CubeX has been designed to be a selling product for interactive/artistic installations. Impressive as well as interesting examples can be seen on the toolkit's website [1].

I-CubeX (see figure 3a), is based on the I-Cube System from Axel Mulder. The target was to develop a data acquisition and processing system for artists to design and create interactive art. Like I-CubeX, this system is based on the MIDI protocol, which is used for the communication between the microcontroller, which is described as "digitizer" within I-CubeX, and a computer or any MIDI device [23].

Equal to all other toolkit's, sensors are used as input devices, but in contrast to them, the data collection is focused on all kind of movements of humans, animals or objects, as well as environmental parameters, like temperature or humidity. Even biological data, like heartbeats or the galvanic skin resistance can be measured. Therefore, the amount of available components might be limited to two domains, but due to the diversity of existing measurable data, the mightiness should not be underestimated.

The setup of the toolkit is not as simple as with Lego Mindstorms. Instead of plugging bricks on top of each other, some wiring is needed, when sensors have to be connected to the digitizer. These can be connected to either MIDI event receiving devices or to a Windows, Mac OS X or Linux system for programming via USB, Bluetooth or MIDI. Supported languages include C and C++, but also Max, an interactive graphical programming environment for music, audio, and media, since it is closely linked with the toolkit's target group.

The power supply is similar to the Mindstorm toolkit, too. Either the digitizer is plugged directly into the power socket, or it is additionally equipped with a battery pack. But in contrast to Lego, I-CubeX has no built-in battery tray.

As being the central device, the digitizer alone is responsible for the data processing and the conversion of the signal from the sensor to MIDI events. Therefore, it is able to accept analog and/or digital signals at any of the maximum 32 inputs and, due to its programming, create an according event, which is sent to a MIDI device, that executes it. Finally, in order to transfer the program from the computer to the digitizer, the above mentioned transfer methods can be used.

Last but not least, it is important to mention, that the I-CubeX is not specifically targeted for prototyping, although it would be suitable, but is more a selling product. Instead, the toolkit is a bit out of the ordinary, since all components are very well designed and have a high quality claim. Furthermore, the modules are easy to integrate and almost configuration-free. However, this comes with a high price. The basic system costs about 330 USD and sensors are between 40 USD and 560(!) USD. Taking everything into account, this is by far the most expensive toolkit, which is covered in this paper [22], [1].

5.3 littleBits

littleBits encourages trial and error, by delivering a playful approach: Little pieces of electronics, which can easily be snapped together.

The littleBits represent a prototyping platform, which literally come near the term plug and play. The basic idea is to have many small and stand-alone entities, which consist of a tiny circuit board in combination with discrete electronic components. These could be a button, a knob, a LED or any kind of sensor and since every entity just inherits one function at a time, this toolkit can be described as a modular

[5]http://brickos.sourceforge.net/

[6]http://www.softintegration.com/docs/ch/

[7]http://www.dcl.hpi.uni-potsdam.de/research/lego.NET/

[8]http://lejos.sourceforge.net/

[9]http://code.google.com/p/nxt-python/

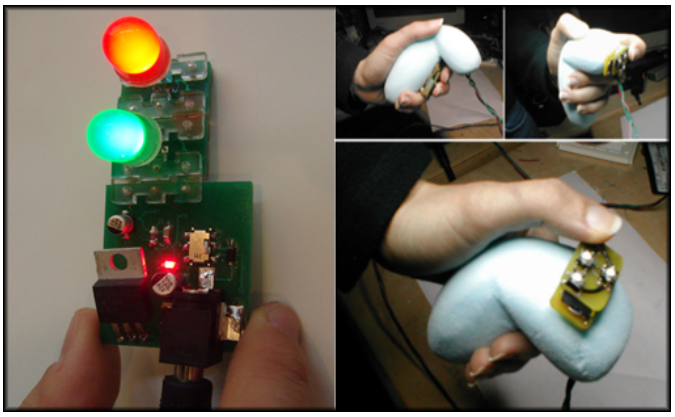


Fig. 4. a) littleBits toolkit example b) Calder Toolkit controller example; Sources: [3], [17]

block system. In order to combine several Bits with each other, Ayah Bdeir and her team designed an easy snap together mechanism, which is based on magnets. A simple example can be seen in figure 4a. When taking up the cause of moving electronics "from the hands of experts, to those of artists, makers and designers" an essential success factor is to push the platform, in order to reach a critical mass of users as well as module developers, which mutually enforce each other. In order to benefit from such a network effect [18] it has been a smart move to make littleBits an open source library.

Nevertheless, since the starter kit has just been released in May 2009, there are unfortunately not many components available, yet. And due to the fact, that there are not many components, besides LEDs, a power supply and a few switches like toggle switches, or sliders available, there are currently not many combination possibilities, which in return limits the mightiness. Furthermore, since all modules are pre-produced and configured, there is no way for the user to alter or to change a module, which is a limiting factor as well.

A fitting example for the high beginner-friendliness is the use of magnets for connecting the Bits: On each edge of a module there is a magnet serving either as power or ground connector, making it easy to stick them together. But additionally the polarity of the magnet is used to enforce the polarity of the connector, making it impossible to connect the modules in a wrong way. The performed user study [8] also underlines this, as participants, who did not have anything to do with electronics or even feared to use them, were really excited about the toolkit.

The power supply is granted through a battery, which can be as easily plugged towards another module, as the littleBits among each other. According to the project's website, there is also a USB connection available, that enables empowering the toolkit.

The location of the processing power cannot be answered easily, since there is actually no microcontroller. All modules just have one function, which is hardwired and cannot be altered. As a result, no complex input or output operations and algorithms can be applied, but it is possible to control the current flow, thus making it possible to dim an LED via a transistor for example.

Supported programming languages as well as platform-independency need no further discussion, because the criterions do not apply to this toolkit. However, this is done intentionally, due to the high beginner-friendly and inspirational claim to work with electronics [8], [3].

5.4 Calder Toolkit

Most toolkits normally focus on the technical design, by delivering a hardware platform. The special thing about the Calder Toolkit is a freely formable foam, into which all components are stuck. This foam also makes it possible to further integrate the esthetic design aspect into prototyping.

The Calder system has been designed at the HCI and School of Design Institute at the Carneige Mellon University. It was specifically developed for product designers, in order to provide them with the capabilities to explore "form and interactivity of product designs"[17] and the ability to iteratively work and evaluate high fidelity prototypes. This toolkit includes a set of input and output modules, which can be reused, a foam infrastructure to connect these wired or wireless components and integration into existing interface prototyping tools.

Considering the available components, it has to be pointed out that the amount is fairly limited. In total, there are just eleven modules, five wireless ones and six, which are attached to wires. In general, the modules are equipped with a microcontroller, which serves as a link to a "global master" (a computer which provides power supply and processing power) and furthermore manages the component's I/O device(s). The wired modules implement a general-purpose input "hub", which has four digital and four analog connections for receiving the corresponding signals. To keep the system simple, these two connector types are incompatible, making it impossible to misconnected devices. Moreover, the toolkit supports "hot plugging", which enables the user to exchange devices during the runtime. For the wireless modules an uplink transceiver is responsible for the communication between the computer and further wireless modules in the vicinity.

On the first sight, due to the limited collection of components, the mightiness could be considered low. Nevertheless, the Calder Toolkit offers an interesting basis for the components: foam. This foam makes it possible for the designer to quickly shape the desired look or make adjustments to it. Moreover, modules are simply pushed into the foam, making it easy to arrange them as well. Consequently, it is not time-consuming to build and evaluate a game controller, like the one in figure 4b.

A few points have already been mentioned, which make this toolkit beginner-friendly: hot-plugging, uni-fitting connectors and modules to stick on freely formable foam. But, additionally the programming infrastructure is easy accessible as well. Via an interactive GUI system it is possible to implement Calder components into a GUI environment, fulfilling the claim the be integrateable into existing systems.

The location of power supply depends on the module in use. A wireless one has its own battery, whereas a wired component receives power via an USB connection. In contrast, the processing power is not provided by the PC, but from the onboard microcontrollers.

The current programming language support is focused on software, that designers use for their visual prototyping, like Macromedia Director[10]. But, as announced by the inventors, the programming languages C/C++, Java and Visual Basic are likely to be supported soon. Therefore, the toolkit can currently only be used on Mac OS X and Windows computers, since Director is running on these platforms. [17].

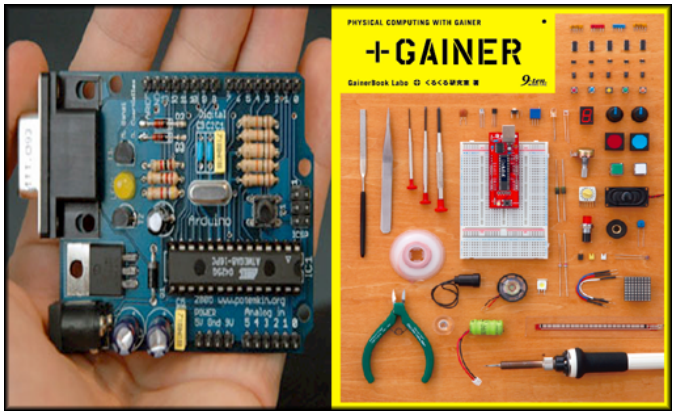


Fig. 5. a) A serial Arduino board b) Gainer toolkit and components; Sources: [10], Shigeru Kobayashi [2]

[10] which is now called Adobe Director

5.5 Arduino

The Arduino board is one of the most successful ones. This may be due to the fact, that it is very cheap in comparison to the other toolkits. At the same time, it offers many additional possibilities with the concept of "shields" as extensions.

Arduino has been developed in collaboration between the Interaction Design Institute Ivrea and the Tisch School of the Arts at New York University, in order to enable designers to express their desired intentions towards engineers for future development. Since it is not likely, that "artists" turn into "techies", or vice versa, a intermediary has to impart between the parties. Arduino, as a prototyping toolkit has been designed to be that link and thereby to enforce and improve the exchange.

One big advantage of Arduino is the quasi unlimited amount of available components, since the board is designed to work with standard electronic ones. Hence, the users can use any sensors or actuators, regardless if they are new or unusual, without having to wait for Arduino fitting versions. Furthermore, "shields" represent extensions to the classic board, which provide extra functionality. RFID readers, Organic light-emitting diode touch displays, Ethernet connectors, Bluetooth interfaces or GPS are just a few examples for modules, that can get just snapped on Arduino and instantly multiply the possibilities and at the same time the mightiness.

Concerning the beginner-friendliness and electronics, Arduino follows an educational approach, by simplifying to the point where users can deal with the topic directly and by themselves. Another major factor is the vast community. On a daily basis, new Arduino projects are presented, shared and discussed by the users and moreover, the countless forums provide an excellent platform for beginners to receive help and inspiration.

As with most of the described toolkits in this paper, internal power supply can be achieved by a battery pack. But in contrast, its integration is easier, since the shield extensions allow an easy snap-on. Additionally, power may also be induced via the build-in USB port.

Each Arduino board (figure 5a) contains a microcontroller, leaving the processing power primarily onboard. Considering the approach from the Calder toolkit, where every input or output component integrated an own small microcontroller, Arduino mirrors a centralized processing approach in comparison to the decentralized one of the Calder modules. However, this is not the only possibility to process data, since Arduino is able to communicate via its serial (USB) port with running programs on computers, using their resources or trigger actions within them.

The Arduino language is based on C/C++ and just includes the "imperative basics". The GUI itself is based on the Processing development environment[11], thus making the SDK easy to use. This SDK has been written in Java and runs under Windows, Mac OS X and Linux, making it platform-independent and able to attract a significant number of developers and persons who want to become one [10].

5.6 Gainer

Gainer stands out by tackling the problem of limited space on a breadboard, by providing so called bridge modules. Complex circuits can be integrated into these modules and therefore save space on a breadboard.

This toolkit is an environment for both education and actual installations. The key concept is, that the user starts with bare components and a breadboard and then builds her or his own I/O module, by soldering the required components. Thereby, the user shall acquire the basics of working with electronics. This "puzzle approach" provides several advantages like easy exchange of single broken components or a very high degree of individualization towards the selection of components. To further support this "handicraft work" the Gainer hard- and software is open source, allowing more advanced users to modify existing hardware to create a new one for their individual project.

The amount of available components is as limitless as with Arduino, since both toolkits follow a similar doctrine. However, the Gainer

[11] http://processing.org/

toolkit (figure 5b) offers one additional feature, which can become very handy, due to the limit space on a breadboard: The bridge modules. These are intended to be combined with the ports of an I/O module, in order to expand the capability of the ports. This is necessary, because as soon as more complicated circuits are needed, the space on a breadboard decreases rapidly. Such circuits are integrated into that bridge component, leaving consequently more space and, depending on the bridge, may provide further values as well. Taking this into account, the Gainer toolkit can be considered even mightier than Arduino, but this clearly shifts the customer focus towards more experienced ones as well.

As a result of this customer focus shift, the beginner-friendliness is reduced, but only on the hardware side. Considering the software however, the user can handle the modules with both graphical as well as code-based programming languages. Support can also be gained from the community, but unfortunately it is mainly active in Japan and Japanese forums, leaving non-speakers out.

Power can be supplied via the USB connection, as well as external power adapters. Interestingly, no battery packs specifically for the Gainer toolkit are currently offered. This is not a big downside, when considering, that this platform follows the same approaches like Arduino and battery packs are easily exchangeable.

The processing power as well follows the Arduino example. But, due to the possibilities bridge components offer, an increase in processing power is theoretically thinkable. One example would be the pre-processing of input data from several sensors at the bridges and then forwarding the results to the central microcontroller, which would only have to process filtered data, for example.

As mentioned above, visual as well as code-based programming languages are supported, which greatly increases usability. On the side of graphical program development Max/MSP is supported by Gainer, providing a good entry point for designers and the code-based side is covered by the Processing environment, which is based on Java. But in between these two extremes a further language can be used for programming purposes: Flash. With these three alternatives, Windows, Max OS X and Linux can be used as a development platform [19].

5.7 "Custom solutions (Barebone)"

The custom solutions fall into the category "Do-it-yourself" (DIY) and become increasingly popular online. Dozens of communities like Makezine[12] or Making Things[13] can be found, where people show and tell, how they tinkered, hacked, reused and (re)assembled components and materials in fun, creative, unexpected or just more efficient ways. Since DIY is strongly connected with individualism and creativity, there might be users who are not satisfied with the offered prototyping toolkits and consequently turn to custom solutions.

Microcontrollers are as well available in so called "barebone configurations". Here, the designer has just the chip and has to do everything by himself, which of course is more work, but leaves him with the liberty to do everything like she or he wants to. Naturally, this leaves the user with an unlimited supply of modules, since these can be bought, hacked, modified or simply self-build. As the mightiness of a toolkit is correlating with the amount of available components, it is obvious, that custom solutions are the most mighty ones to work with.

Nevertheless, the mightiness comes with a price: Barebone configurations are very far from being beginner-friendly and are only suitable for experienced users. For example, it is up to the designer to build the microcontroller supporting circuitry, which is a quite complex task.

Considering the location of power supply and processing, it is again completely up to the user how it is to be implemented and what is to be implemented. As a result, this stresses even more the high mightiness, but as well the very low beginner-friendliness.

Microcontroller programs are usually made with the Assembler language. On the one hand, this might not be the easiest language to program in, but on the other hand it is platform-independent, since a simple text editor represents a sufficient development environment.

[12] http://www.makezine.com/

[13] http://www.makingthings.com/

Fig. 6. Paper prototyping examples; Source: [20]

It cannot be emphasized enough, that this is the most challenging and maybe also frustrating approach, however at the same time, the user has almost unlimited possibilities at her, or his hands [22], [9].

6 Special Toolkits

To round things up, this paper will shortly introduce two special kinds of toolkits, which deal with paper computing and mobile prototyping. These approaches stand for current and interesting developments within the fields of physical computing, prototyping and hardware sketching.

6.1 Paper computing

Paper computing stands for an innovative approach which combines programming, painting and papercraft, which allows users to create functional prototypes set on painted paper. On these grounds the MIT Media Lab in collaboration with the Craft Technology Group introduced a toolkit, that consists of microcontrollers, sensors, actuators and power sources, which are fixed on paper surfaces via magnets. Besides its magnetic characteristics, the paint has conductive ones as well and is therefore used as "wires" between the modules, which can be moved freely on the surface, like magnets on a whiteboard. Two examples can be seen in figure 6.

The aspect of available components poses a ambivalent issue. In general, every imaginable module could be used, but it has to be prepared in a complex process before it is ready to be used in this toolkit. Each module is glued on uniquely colored (i.e. switches are green, LEDs are pink, ... , etc.), magnetic and conductive paper and its polarity is indicated via the paper's shape (flat sides are negative, rounded ones are positive). Furthermore magnets are attached, in order to enable the component to stick to the big paper surface. Since normal wires are no adequate way to connect the modules in this toolkit, special conductor paths have to be used as well. In this case, CuPro-Cote[14] paint has been used, which has the advantage, that it can be used like normal water-based acrylic paint.

The mightiness should be seen differentiated as well. Since the amount of components "out of the box" is limited, the factor would initially be medium to rather low. Nevertheless, it is possible to modify "normal" components for the use with the toolkit, which increases the factor again.

The beginner-friendliness of this toolkit is obviously very high. It is very easy to just draw the circuits and snap the modules as needed onto the paper. One minor problem could be the removal of the painted lines, which is more difficult than just pulling a wire.

The locations of power supply and processing is, fitting to the artistic claim, integrated into the "artwork". Batteries as well as microcontrollers are simply snapped onto the surface. Considering the external

[14] http://www.lessemf.com/292.html

supply with power sources or processing capabilities via an USB connection, no concrete implementation could be found, but should not be a difficult task to develop.

The microcontrollers are programmed by using the Arduino IDE, which consequently requires some knowledge in C/C++ and provides the same high platform-independency by running on Windows, Mac OS X and Linux systems. In order to program a picture/prototype it is possible to connect its 4 pin header with a USB port of a computer and the upload is realized via the Arduino IDE.

Especially for this toolkit a new criteria has to be introduced: awesomeness. It is simply amazing how easy it is to paint, rather than "build" the prototype. Not to mention to process of painting itself, which is a counterpart to the classic approach.

Last but not least, this toolkit also provides new and interesting ideas for interactive paintings, new kinds of games or teaching [20].

6.2 iStuff mobile

iStuff mobile bridges the gap between hardware and software prototyping, by delivering a framework, that allows the use of external sensors and actuators in combination with standard mobile phones, which do not have to be modified in order to be used within this toolkit. Thus it is possible, to tackle the problem to integrate new hard- and software without making internal modifications to the handset, which is a large obstacle for rapid prototyping and researchers. Due to the fact, that this toolkit represents an "in-betweener", the above introduced advisory matrix is not completely fitting and therefore, iStuff mobile will be generally described in order to round up the toolkit presentation.

This toolkit is based on hard- and software prototyping environments. For the hardware aspect Ballagas et al. focused on the physical hardware toolkit Smart-Its [11], which provides the sensor network platform. The underlying advantage of Smart-Its is, that it allows the reconfiguration of the sensors via a procedure call interface and therefore applies these changes without the user having to reprogram the sensor boards. On the software side, the iStuff [7] framework in combination with the Event Heap [16] and Apple's Quartz Composer[15] builds the foundation for development.

The general setup for this toolkit is a Smart-Its sensor board in combination with a Bluetooth communication device, which are attached to the back of a phone. As soon as the sensor registers a movement for example, the data is transmitted wirelessly to the Smart-Its x-bridge, which bridges between the wireless devices and a local software infrastructure. One important part is the Smart-Its Proxy which collects the data and encapsulates it in Smart-Its events, which are subsequently forwarded to the Event Heap, which is the second part. This approach covers the input via the sensors, but in order to process input from the mobile phone a further channel is needed. This is provided by a running application on the phone, which intercepts key presses for example and passes them on via the build-in Bluetooth module inside the phone to a third part of the software infrastructure, the Mobile Phone Proxy, which encapsulates the data into iStuffMobile Events and pushes these onto the Event Heap as well. When it comes to processing these events, the Quartz Composer in combination with special plugins for the iStuff's Patch Panel, as a forth and final part, is responsible for transforming the input events into desired output events, like a display device for example. For output on the mobile phone, the Phone Proxy listens for iStuffMobile Events on the heap and forwards them via Bluetooth to the application on the mobile phone. Receiving these events as well as sending events like the mentioned key presses are handled by a background application. This application either executes the command directly or passes it on to a foreground application, which deals with the direct user interaction [27].

Obviously, this toolkit is not suitable for random places, but requires an existent infrastructure. Therefore it is best suited for instrumented environments or generally in the field of ubiquitous computing and it demonstrates a way to easily create sensor-enabled applications on mobile phones. However, this toolkit seems a bit outdated in comparison to Apple's iPhone, which is able to provide own sensor data as well as the means to process the input internally. Nevertheless, since the iPhone is limited to three sensors, an interesting approach would be the extension with further ones in combination with this toolkit.

[15] http://developer.apple.com/graphicsimaging/quartzcomposer/

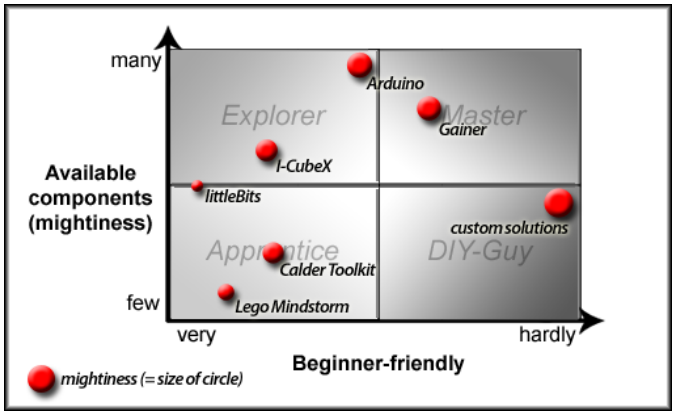


Fig. 7. Toolkit advisor framework; Source: own illustration

7 Discussion

Now, with the advisory framework established and a few exemplary toolkits introduced, it is time to put both together and rank the toolkits inside the matrix. Thereby, recommendations will be created, which toolkits fit best to the according profiles. Besides the criterions of the axis, other differentiation ones will be included as well, in order to ensure the best user fit possible. The complete matrix with the assigned toolkits can be seen in figure 7 and will be discussed below. For this figure the framework has been extended with a further dimension in order to display the mightiness of the toolkits.

The "Apprentice" has not yet gathered much experience with programming, soldering or electronics in general and is looking for an easy introduction. Therefore the adequate toolkits are characterized with a high beginner-friendliness in combination with just a few available components to reduce complexity.

The first fitting example is Lego Mindstorm, which offers the fewest modules and is very easy to use. The reasons for that are versatile. First of all, the use of the Lego bricks metaphor, which makes it much more intuitive for the user, because she or he is already used to stick the bricks together. Secondly, the native support of visual programming, and especially the user-friendly LabVIEW, makes programming basically a drag and drop task and allows the quick and easy creation of programs. Last but not least, the toolkit provides with the NXT a central element which integrates processing power and electricity supply, which has the advantage, that the user does not have to worry about the right plugging of the modules. This as well makes it easier and allows focusing on the project.

The second suitable toolkit for the Apprentice is the Calder Toolkit. It delivers a few more components than Lego Mindstorm, and can be considered a bit less beginner-friendly, since the user gets in contact with electronics for the first time. Nevertheless, hot-plugging, unifitting connectors, surrogate objects for programming and the ability to stick modules freely on formable foam makes this toolkit still a great recommendation for starters. But there is a little catch: currently, only Macintosh and Windows users are able to program for that toolkit, since Linux/Unix systems are not supported by the Adobe Director, which is used for software development.

The littleBits are stuck in the middle between "Apprentice" and the "Explorer", due to their limited, but greater supply with components and the even higher beginner-friendliness in comparison to Calder and even Lego. Therefore this toolkit could be seen as a "transitional" one between the two profiles. The unique selling point of the littleBits is, to make prototyping a bit like puzzling. All the single components stand alone for themselves and can simply be snapped together with

other modules. As a result, no programming languages have to be learned as well, since the components are autarkic and basically just control the power flow via buttons, sensors or knobs. This whole setup makes this toolkit probably the most simple one in this paper, but consequently does not offer any complexity or the possibility to process input or output operations. This case shows, that it is essential, to provide further criterions or dimensions in the framework, since littleBits would be a better choice than Lego Mindstorm at the first sight. By providing further information about the mightiness, the user is better supported in determining the right framework for her or him.

I-CubeX falls completely into the "Explorer" category. The crucial point of this profile is, that beginner-friendliness is still important, but at the same time more components are demanded by the user. This is, because the modules provided by Apprentice toolkits are not up to the task anymore, to fulfill the Explorer's demands. I-CubeX fits into this category, which is due to the relatively high amount of available components, which is paired with still high beginner-friendliness. The high usability is granted through almost configuration-free and easy to integrate modules as well as the support of Max, a graphical programming environment. Although, "only" MIDI events are exchanged, which on the one hand can limit the mightiness, it should not be underestimated on the other hand, since a wide range of movements and environmental data can be measured and processed. Furthermore, the digitizer supports input from 32 different sensors, which enables complex setups. A nice plus are the well designed components which suffice even fixed installations.

Putting the Arduino toolkit inside the "Explorer" category, may be a bit underestimated. Although it is still easy to program via the Arduino programming environment and easy to use with the "plug-in" wiring of sensors, the huge amount of available components makes this toolkit very popular and powerful. Here, the developers found a great trade-off and this may also be the reason why "Masters" often choose Arduino for their projects. Furthermore, this toolkit is very cheap compared to other platforms and introduced the concept of shields, making it possible to extend an Arduino with complex features, like a small, touch sensitive organic light-emitting diode display. Obviously, this is also a major factor, which raises the mightiness even more. Last but not least, the vast Arduino community is contributing greatly to the toolkit's success. Here, the users exchange project ideas and solutions, help each other and brainstorm for new concepts. Moreover, source codes and circuit plans from projects can be downloaded and reverse engineered by Apprentices to learn more.

Just like the "Explorer" profile, the "Master" still draws on many available components, but is not anymore constrained to high level programming languages and made already some experiences with electronics and soldering. The Gainer toolkit is a fitting example for this category. The beginner-friendliness on the hardware side is quite low, since the user has to build and solder her or his own I/O modules. Having done so, the microcontroller, breadboard and I/O have to be puzzled together. However, this is not a bug, but a feature, since the developers designed the toolkit in this way on purpose, in order to get the user more acquired with electronics. However, the software side is much more easy to use, since Flash, Max/MSP and Processing is supported for programming. The amount of available components and consequently the mightiness is slightly smaller than with Arduino, because Gainer has no extension option via shields. A further minor subtraction has to be considered with the beginner-friendliness for users, who do not speak Japanese, since the majority of the community is situated in Japan and the forums and web pages are often just in Japanese language.

The "DIY-Guy" as well as the custom solutions play a special role. The user profile is characterized by the low reliance on premanufactured modules and as a result, everything has to be made by oneself. Naturally, "handicraft" is a long and expensive process, which is not suitable for commercial product development. Concerning the toolkit, it is obvious, that the beginner-friendliness is nonexistent, since all modules are made by oneself. At the same time the amount of available components seems to be unlimited on the first sight. But it has to be taken into consideration, that the amount of available modules is correlated with the individual skills in programming and electronics. This is due to the fact, that for every new module a certain amount of knowledge and experienced is required and the more complex the component gets, the more time consuming and demanding is the construction and assembly. In order to reflect this fact in the matrix, this "toolkit" will be put close to the "Master" border and the indication of mightiness is set to a high value.

8 Conclusion

Product development is an essential part of many companies [25], and there are a lot of factors, which have to be considered. First of all, since it is always about the money, it is important to meet the budget. As soon as the development costs are too high, the product has to be sold for a higher price, which can have a great impact on sales. Secondly, product evaluation is important, because the customer has to like the final product. The third point is closely connected, because a late change in the product is much more expansive, than many little changes within the development. Therefore, product development and evaluation have to be iterative. Due to the fast-paced technology world, every day new innovations appear and have to be taken into consideration of the own product. This is the reason for the forth demand, which is modularity. By having different modules, it is easier to exchange it with another, and not having to redesign everything. Closely connected to this is the reusability of modules, which is not only environmental friendly but also more cost effective. Furthermore, speed, as the sixth factor is also a strong demand, considering the fast technology world. This list of demands is not exhaustive, but mirrors some important factors, which prototyping toolkits and hardware sketches are able to tackle. Already with the introduction of prototyping, the product development costs can be significantly reduced [6]. Secondly, prototypes represent a handy tool in order to collect customer feedback on the product, to lower the "non-acceptance risk" from the customers. Additionally, hardware sketches allow an interactive evaluation during the product development and therefore fulfill the third demand as well [13]. Modularity and reusability can also be achieved, due to the fact, that toolkit prototypes a priori are often separate input and output modules, which are easy exchangeable [26]. According to Backhaus et. al [13] rapid prototyping also fulfills the last demand, which is speed. Many toolkits serve that need as well, since they provide an easy way to quickly combine different modules. By outlining these demands, it becomes clear, that prototyping tools are an essential part as well as a vital tool of and for product development.

This might also be one of many reasons for the wide variety of hardware sketching toolkits. But besides this commercial approach, there are many more: science, education, hobby, entertainment and so on. Due to this high diffusion rate, it is very difficult to keep track with new and current toolkits. Furthermore it becomes difficult as well to compare the toolkits with each other and give concrete recommendations, which toolkit is suitable for whom. In order to contribute to the solution of this problem in the future, this paper presented a toolkit advisory framework, which is able to support a user in his search for a fitting toolkit via user profiles. Since every profile inherits many platforms, further differentiation criterions can be applied, in order to reduce the number of options. This framework takes less time and is more effective than conventional research. For starters, a few toolkits have already been analyzed and ranked in this matrix, in order to provide an entry point for future toolkit analysis as well as selection. Finally, the author hopes, that this framework might help to introduce many more people to the exciting world of hardware sketching.

References

[1] About I-CubeX. http://infusionsystems.com/catalog/info_pages.php?pages_id=117, last accessed on 01.12.09.

[2] Gainer Exhibition. http://gainer.cc/Exhibition/PlusGainer, last accessed on 23.12.09.

[3] littleBits. http://www.littlebits.cc/index.html, last accessed on 03.12.09.

[4] What is NXT? http://mindstorms.lego.com/en-us/whatisnxt/default.aspx, last accessed on 01.12.09.

[5] *Reflective Physical Prototyping through Integrated Design, Test, and Analysis*. ACM, October 2006.

[6] 3D Systems. Product development at laser speed - getting to market first. http://www.metrorp.com/images/ProductDevelopmentAtLaserSpeed.pdf, last accessed on 10.12.2009.

[7] R. Ballagas, M. Ringel, M. Stone1, and J. Borchers. istuff: A physical user interface toolkit for ubiquitous computing environments. In *CHI 2003*, 2003.

[8] A. Bdeir. Electronics as material: littlebits. In *Proceedings of the Third International Conference on Tangible and Embedded Interaction (TEI'09)*, 2009.

[9] L. Buechley, E. Paulos, D. Rosner, and A. Williams. Diy for chi: Methods, communities, and values of reuse and customization. In *CHI 2009*, 2009.

[10] D. C. David A. Mellis, Massimo Banzi and T. Igoe. Arduino: An open electronics prototyping platform. In *CHI*, 2007.

[11] H. Gellersen, K. G., A. Schmidt, and M. Beigl. Physical prototyping with smart-its. In *IEEE Pervasive Computing 3*, pages 74 – 82, 2004.

[12] S. Gregor. Physical interaction design. Seminararbeit, February 2009.

[13] P. D. D. h.c. Klaus Backhaus and P. D. M. Voeth. *Industriegtermarketing*. Verlag Franz Vahlen Mnchen, 8. auflage edition, 2007.

[14] L. E. Holmquist. Sketching in hardware. *interactions*, 13(1):47 – 60, January + February 2006.

[15] J. W. Jason Gilder, Michael Peterson and T. Doom. A versatile tool for student projects: an asm programming language for the lego mindstorm. *Journal on Educational Resources in Computing (JERIC)*, 3:Article no. 2, 2003.

[16] B. Johanson and A. Fox. The event heap: A coordination infrastructure for interactive workspaces. In *WMCSA '02: Proceedings of the Forth IEEE Workshop on Mobile Computin Systems and Applications*, 2008.

[17] S. E. H. J. F. P. H. D. Johnny C. Lee, Daniel Avrahami and D. Leigh. The calder toolkit: Wired and wireless components for rapidly prototyping interactive devices. In *Designing Interactive Systems, Dis 2004*, 2004.

[18] M. L. Katz and C. Shapiro. Network externalities, competition, and compatibility. *The American Economic Review*, 75:424–440, 1985.

[19] S. Kobayashi, T. Endo, K. Harada, and S. Oishi. Gainer: a reconfigurable i/o module and software libraries for education. In *NIME '06: Proceedings of the 2006 conference on New interfaces for musical expression*, pages 346–351, Paris, France, France, 2006. IRCAM — Centre Pompidou.

[20] S. H. Leah Buechley and M. Eisenberg. Paints, paper, and programs: First steps toward the computational sketchbook. In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, 2009.

[21] C. Moussette. Tangible interaction toolkits for designers.

[22] C. Moussette. Tangible interaction toolkits for designers (short).

[23] A. Mulder. The i-cube system: moving towards sensor technology for artists. 1995.

[24] S. O. Onuh and Y. Y. Yusuf. Rapid prototyping technology: applications and benefits for rapid product development. *Journal of Intelligent Manufacturing*, 10:301 – 311, 1999.

[25] M. E. Porter. *Competitive Advantage: Creating and Sustaining Superior Performance*. The Free Press, 1985.

[26] J. Portilla, A. D. Castro, A. Abril, and T. Riesgo. Rapid prototyping for multi-application sensor networking. *SPIE Newsroom*, page 1, 2007.

[27] R. R. Rafael Ballagas, Faraz Memon and J. Borchers. istuff mobile: Rapidly prototyping new mobile phone interfaces for ubiquitous computing. In *CHI 2007 Proceedings*, 2007.

[28] B. von Oetinger. *Das Boston Consulting Group Strategie-Buch: Die wichtigsten Managementkonzepte fr den Praktiker*. Econ, 2000.

[29] E. v. Weizsäcker. Erstmaligkeit und bestätigung als komponenten der pragmatischen information. *Offene Systeme I Beiträge zur Zeitstruktur, Entropie und Evolution*, 1, 1974.

Prototyping in Physical Computing - Sketching in Hardware

Thomas Bauer

Abstract— Hardware toolkits are gaining more and more attention, not least from the sketching and prototyping point of view. Ideas can be tested, evaluated and thus improved in a very early stage which can save a lot of time and energy. In this seminar paper I will try to give an overview on some frameworks and analyze them in terms of their level of abstraction and the possibilities they offer consequentually. After that a closer look at a number of concrete implementations is taken to further point out the difference between the distinct systems before an outlook on an evaluation is given.

Index Terms—Prototyping, Hardware, Toolkits, Ubiquitous Computing, Tangible Interfaces, Mockups, Sketching

1 Introduction

The interface between human users and the computer is one of the major problems in Information Technology at the moment [29]. Ongoing research in the field of Human Computer Interaction (HCI) and Human Computer Technology (HCT) is adressing this task and tries to narrow the existing bottleneck that hinders the high performances possible within the human mind to get through directly to the machine that can process it. Thus more powerful interfaces are being created in many different variations and forms. The goal is to augment the overall efficiency when interacting with a computer. Prototyping is an essential part of designing such concepts [3] as evaluation can take place in very early stages. Problems can be identified, positive tendencies further explored. And all that without any risk as such schemes or drafts are usually rather inexpensive and fast to develop.

There are many different methods of prototyping graphical user interfaces (GUIs). Paper prototypes [8] that work without any digital help by sketching drafts on paper and cardboard. Wizard of Oz [9] setups, where functionality is imitated by a hidden person in the background. Or Mockups [10], rudimentary pre-versions of interfaces and applications that dont offer functionality but the possibility to browse and explore the interface. There are more options and variations but it is noticeable that all of those prototype options offer possibilities for fast, easy and very low-level realization - if desired.

When observing current developments in HCI/HCT a noteable growth of tangible devices to interact with the digital world can be experienced [5] [7]. Generally speaking, more channels to transmit information get opened in both directions. Speech control systems for example gain more and more attention by users and developers, ending the unidirectional flow of audio information. Other examples include touchscreens (iPhone, Microsoft Surface) or move-sensitive devices (Nintendo Wii, iPhone). More abstract projects are being developed such as motion sensored installations or devices with haptic feedback. Some of them have found their way into the commercial world, more and more will in the near future.

The easy access to computers and software developer tools has proven to be a hotbed for new ideas concerning software applications. The same effects can be expected on the hardware side. It is nearly impossible for a single person to understand all the details of an ordinary mp3 player. Only when put on a higher level of abstraction lay people that come from different backgrounds will have the chance to work and actually implement their ideas hardware-wise [6] [1]. In our context this is exactly the case as computer scientists who do research on the digital world often reach their limits when facing more hardware-oriented problems and tasks. Such toolkits are hence heavily needed and imply an important part within the more hardware-directed fields of computer science like Ubiquitous Computing and Human Computer Technology [4]. Toolkits can help to create first mock ups of systems, but they can also serve as a complete construction kit. This depends on the final goal and the kit used. Examples will be discussed in section 4. To get a complete first overview on existing hardware frameworks see [2] and [19]. For this study seven representative toolkits were picked.

The main contribution in this paper is to give the reader a first insight in the subject, a basis to further explore the field and to support the search for the framework that seems to fit his/her level of hardware knowledge the best. Concrete examples will further recess the picture.

2 Interesting Hardware Toolkits: An Overview

Seven toolkits will be introduced in this section. The intention is to cover a broad field of different approaches. Every implementation has its own focus and goals and I will try to point them out properly. To start things off let me first talk about all the toolkits covered separately.

2.1 Arduino

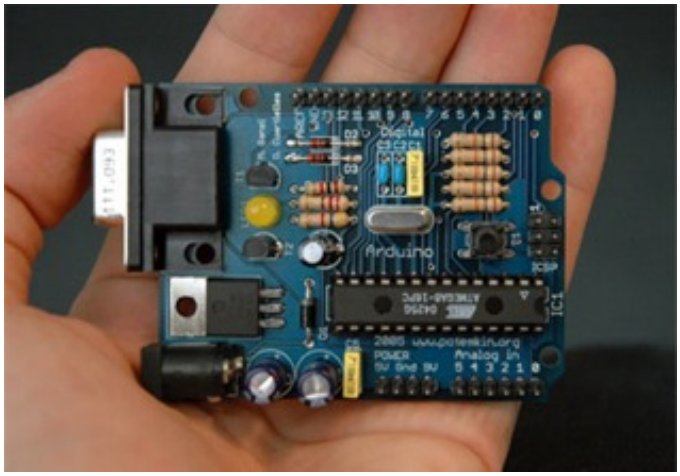


Fig. 1. Arduino basic module. [12]

The intention behind the Arduino [12] project is to provide a little minicomputer (fig. 1) that is strictly free of barriers, that is open source. Thus there are various different versions available: Mega Board, Duemilanove, Mini, Nano, Pro, and many more. As very basic hardware components are used ('open hardware') those boards can

- *Thomas Bauer is studying Media Informatics at the University of Munich, Germany, E-mail: bauerth@cip.ifi.lmu.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009/2010.*

be reproduced by anyone with proper electrical engineering knowledge. In general they all share property of being a little computer that can process different in- and outputs. Linked to an usual personal computer functions and actions can be added and edited on the board directly. The provided software also allows for scripting on a higher level where C and C++ can be used. Proper documentation is available and a noteable community has grown around this platform. In combination with additional sensors and output devices this approach can offer a wide range of possibilities.

2.2 BASIC Stamps

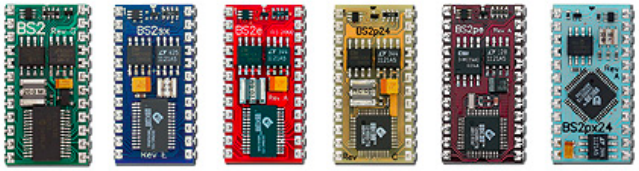


Fig. 2. BASIC Stamps: tiny components, tiny devices. [13]

BASIC Stamps [13] are tiny computer modules that are to be programmed in BASIC. Again, there are various different versions of stamps (fig. 2) for all kinds of different purposes. Commercial interests seem to play a bigger role here as there are many different manufactuers offering and selling correlative products. On the technical side though they all share the fact of being really tiny. Different I/O slots are given to communcate with either computer or other devices or stamps. One of the more popular versions from Parallax also offers a socket to enable communication with environments like Flash or Max/MSP. Documentation is available in detail. In the end a lot of the more detailed specification depends on the company the Stamps are bought from.

2.3 littleBits

littleBits [14] is a promising project with a slightly different approach. It offers a set of preassembled hardware components that can be linked to each other via magnets, just like Lego bricks. The bits are divided into four kinds of types. Power bits provide electricity. Wire bits include wires or logic circuits. Input bits offer a wide range of switches, buttons and sensors while output bits include LEDs, motors, displays and speakers. This toolkit is not yet commercially available, a preorder can be placed at the website [11]. Overall, simplicity plays an important role within this project. The open source approach certainly unlocks barriers but using this framework it is hard to think of the realisation of more complex projects, e.g. projects that require a certain amount of computing or logic.

2.4 Electronic Brick

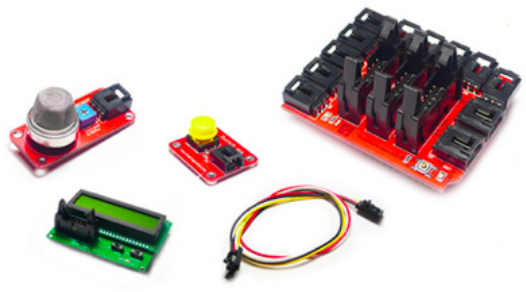


Fig. 3. Electronic Brick: Robust prototyping paired with a huge variety of modules. [15]

Electronic Brick [15] by Seeedstudio is another representative I want to talk about. This extensive framework offers a variety of different low-level modules that can be linked via 3 or 8 wire bus cables using a central MCU unit. Seeedstudio provides such a unit although any Arduino compatible board can be used. Bricks in the commercially available starter kit (in some parts shown in fig. 3) include an angle sensor, a light sensor, a buzzer and a 16x2 LCD display. A robust design and the prototyping perspective are pointed out in the toolkit's description. As Arduino boards can be used as the main hub the software side works accordingly to the first example covered in this section.

2.5 Phidgets

Another popular toolkit is Phidgets [16] [5] [6]. Having had a more scientific background in the past this extensive library of small-sized hardware pieces is now available commercially through a spin-off called Phidgets Inc. The initial intention was derivated from (desktop) widgets that allow for easy creation and customization of graphical user interfaces (GUIs). This concept was ment to be put into a hardware context. Developed at the University of Calgary in Canada, Phidgets offer a wide range of possibilities for a broad spectrum of users. Modules are plug and play and many different programming languages can be used on the software side using sockets (C, COM, Java, .NET, Flash). Unlike other toolkits it is not possible to run Phidget projects in standalone mode as the created devices have to be linked to a computer at all times.

2.6 LEGO Mindstorms

Having talked about 'Lego' earlier, here it is: the real LEGO toolkit [17]. Definately not built with the intention to provide hardware bits for physical prototyping this product offers all the neccessarities to perform such tasks, anyway. Numerous I/O devices are provided such as sensors, buttons, controllers and audio units. Those devices can be connected through a computer the prototyper can program using the provided graphical software. Connection to a host computer is possible via USB and Bluetooth.

2.7 The BUG

Fig. 4. The BUG: High-Level prototyping with class. [18]

A little bit like grown up's Lego seems bug lab's The BUG [18]. Consinsting of a base module and a small number of I/O modules (fig. 4) that can be clipped on to it this solution is keeping it simple and clear. The fully programmable computer in the base module can be accessed via USB, Wifi, Bluetooth and Ethernet. Java is used as programming language. The available modules include a GPS locator, a sound module with speaker and microphone, a touch-display and a motion detector. The simplicity of this product leads to the fact that it is designed rather bold and unflexible.

Table 1. Toolkits: Pricing

Toolkit	Price
Arduino	30USD for Base Module (no sensors)
BASIC Stamp	40USD for a stamp module (no sensors)
littleBits	Not yet specified
Electronic Brick	60USD for starter kit
Phidgets	80USD for basic interface kit
LEGO Mindstorms	400USD for basic kit
The BUG	750USD for BUG Bundle

Table 2. Toolkits: Programming Languages

Toolkit	Available Programming Languages
Arduino	Open Source Environment (+more through Proxies)
BASIC Stamp	BASIC (+more through socket server)
littleBits	Physical options, presets
Electronic Brick	Own basic language (docs available)
Phidgets	C, COM, .NET, Java, Flash, MAX/MSP..
LEGO Mindstorms	Proprietary Visual Editor
The BUG	Java

3 Attempt of a Classification: Hig Level vs. Low Level Solutions

Besides the seven toolkits chosen there are many more options to pick from. Those include among others kits like the Beagle Board [24], The Create USB Interface (CUI) [25], I-CubeX [26], Make Controller [27] or the NerdKits [28]. The variations inbetween all of those systems point out the fact that at the end of the day any related aggregation of electrical pieces can be a hardware toolkit. It is a matter of the user, what he can think of and what he can create.

Nevertheless I will try to put the presented frameworks into relation. As this paper is ment to be a help especially for the beginners on the field of hardware prototyping I will put a special focus on the level of abstraction. How close are the toolkit's modules to the very basic electro-technical elements? This is of great importance since it will in most cases determine how experienced a user has to be in order to reach his goals. In most cases it will also affect the price and the level of detail that can be worked in a crucial way.

3.1 High Level: The straight Legos

The most beginner-friendly possibility is obviously the LEGO Mindstorms set (fig. 5). It is designed for kids and it seems rather implausible that a deep understanding of electrical or software engineering is required. More likely the intention was to offer a product that provides the possibility to play and learn in an engineering context. This of course doesn't mean Mindstorms is not a toolkit that can be used to strive for higher goals. Pieces can be combined in any possible way and the parts itself are of high quality and oriented towards the functionalities the other frameworks are offering. All in all the bold design of the modules and the fact that no real programming is possible (as opposed to other toolkits, see table 2) leads to the conclusion that possibilities are somewhat limited in a prototyping context, especially when including external elements. Nevertheless this toolkit offers a way of putting simpe ideas into reality which can serve as a starting point for a more detailed implementation later on.

Fig. 5. LEGO Mindstorms: Powerful children's toys. [17]

Having talked about LEGO Mindstorms and going on with the BUG toolkit now it seems a little bit like leaving the McDonald's kid's paradise to go next door for the classy french restaurant. It's not the functionalities offered that make this comparison possible, it's the general design and approach to the subject. BUG modules have a very unique and smart look. Besides this aspect there are a lot of similarities: a base module, easy putting together of the pieces and a high technical standard. The fact that the BUG can be programmed in Java is probably the biggest difference the two Lego toolkits can be divided into. As Java itself can be seen on a relatively high level of abstraction this framework is still a very beginner friendly one. And with the breakout board, a module that allows to link external devices via USB, the BUG universe seems wide open again.

3.2 On A Medium Level: The All-Rounders

In this section I will talk about toolkits that can make it all possible, but that are not so consumer oriented and closer to the basic levels of hardware. Phidgets (fig. 6) and Electronic Bricks (let me include Arduino here) are two commercially available frameworks that stand out especially because of their very wide range of modules they offer. Where other toolkits (if you can use that term with those at all) consist only of an I/O board these two models offer already four. And when it comes to the actual pieces to play with possibilities seem endless: distance/range or force/pressure sensors, all sorts of motors, buttons, switches, relays and more - see table 3 and table 4 for an overview on in- and external linking possibilities. Components can still be connected to each other easily via USB or BUS cables, and the fact that their buildup is rather rudimentary keeps the price to be payed on a fair level. Compared to the lego kits pricing can be put into a range of a fifth to a tenth. More detailed information about costs can be found in table 1.

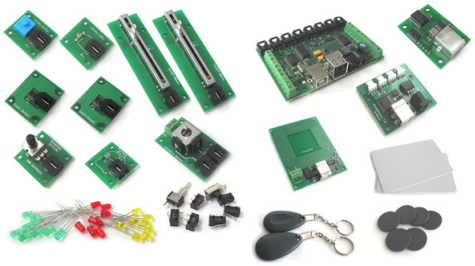


Fig. 6. Phidgets: Wirde range of possibilities, easy connectivity. [16]

The software part with those tools might lay out the biggest obstacle for most of the beginners. Electronic Bricks' open source approach with given applications and an onboard C-like programming environment surely will ask for some time before the user can get familiar with it. The fact that there is another, more user-friendly and Java-oriented programming environment provided for the host computer is definately a big plus. Phidgets socket approach permits even more diversity on the software side. The biggest difference between the two projects in this section might be the lack of a processor module in the Phidgets approach. Summing up one can say this paragraph consists of toolkits that are set up in a very complete way, that are rather small and rudimentary and that require certrain programming skills.

No hardware knowledge is used whatsoever, so these projects might be interesting to look at for the software people.

3.3 Low Level Bits: The Little Ones

The reason why the next two frameworks were put into one category is very simple: they are really small. BASIC Stamps, as the name already states, have basically the size of a stamp. A usual toolkit consists of a programmable microcontroller, cables and LEDs and depending on the order extra features like pushbuttons, a potentiometer or a piezospeaker. The littleBits toolkit's components (fig. 7) are as tiny as the BASIC Stamp's. They seem a little more user-friendly because of their magnetic linking possibilities. A big questionmark poses the nonexistence of a main module, all the bits are preassembled. The final release has still to be awaited but the project looks really promising as it seems to combine the lego approach with very low-level building blocks.

Fig. 7. littleBits: Low-Level Prototyping made easy. [11]

Overall it is obvious that the small size has to result in certrain trade-offs. And this is not even mainly the hardware side where most important functionalities, especially with the BASIC Stamps, are provided. But the software side can form a major obstacle for people that are not familiar with writing computercode on very low levels. With littleBits it's not entirely possible to say, but the lack of a programming environment naturally trims the number of possibilities. To guarantee a high variety of project options the designers aimed for an open source strategy. This is supposed to result in a high count of components. It will be interesting to see if such an approach can work out and where it reaches its limits.

As for the littleBits I want to note that it seems plausible to put them into the first group, the Legos, as well. It shares the characteristic of easy connectivity without the need of a deep understanding of the different pieces. Still it is an approach on a very low level, very close to the actual elements used in electrical engineering. Sharing this fact with the BASIC Stamps in many ways I decided to put them into this section.

4 Example Projects

In this section I will present example projects from all the three levels of abstraction I just talked about. This will point out the different aims these toolkits were developed for and further help the reader to find the category his projects might fit in the best.

4.1 The BUG: It actually has Apps

When looking at bug labs' website the first link in the menu after 'the BUG' is called 'Apps'. When going through this list which, at the moment, consists of 164 entries, one can clearly see where this project might be going: an open source iPhone, without the phone of course.

Table 3. Toolkits: Module to Module Interfaces

Toolkit	Computer Interfaces
Arduino	Serial port, USB
BASIC Stamp	Serial port, USB
littleBits	-
Electronic Brick	User defined
Phidgets	USB
LEGO Mindstorms	USB, Bluetooth
The BUG	USB, Wifi, Bluetooth, Ethernet

Table 4. Toolkits: Device to Computer Interfaces

Toolkit	Hardware Interfaces
Arduino	11 I/O interface pins
BASIC Stamp	16 I/O interface pins (typical)
littleBits	Magnets, Undefined possibilitiets for Extensions
Electronic Brick	Pins Slots
Phidgets	Specified to each module, specialized connectors
LEGO Mindstorms	4 inputs for sensors, 3 outputs for motors
The BUG	4 Bug Module Connectors, USB for external

One could also call it a fully customizable iPod Touch. Hardware can be put together as desired (and extended via USB), software can be put on the main computer hub and managed with the help of a preinstalled Linux operating system. All in all this has led to a number of community applications like a GpsLogger for Google Maps, a Barcode Scanner or a Twitter application. It will be interesting to see if the fact that the hardware is customizable can lead to any major advantages as opposed to devices that strive for having all functionalities included and embedded in one closed up system. From a (hardware) prototyping point of view on the one hand it is clear that the fixed design of the device prevents innovation in a way since possibilities seem limited. On the other hand the BUG can be seen as a functioning system that only needs to be programmed. In this regard the user gets more or less total freedom in what he creates.

4.2 Phidgets: Everything is possible

With Phidgets being a very extensive toolkit any idea seems to be possible. The relatively small and basic pieces can be put in the background, functionality and individual design ideas get more important. There are hundreds of example projects available online, they can be browsed from the Phidgets website.

An example of work possible with Phidgets is the Antique Weather Clock [21] (Fig. 8). Using an old antique clock with a fake clock face it is capable of showing the current weather situation in real-time. To do so it connects to the internet and gets the needed information from a webserver using a little Flash application. After processing the data it puts it into one of the twelve states available on the clock as a little motor moves the needle into the right position.

Another interesting widget [22] was created by Liz Friesen at the University of Calgary in the iLab where the Phidgets framework was developed. She linked a small toy dog with an instant messenger application. Activity in the instant messenger application caused the dog to move around, to notify the user. Additionally the dog moves when there is movement in the physical room. This happends through the use of a motion sensor. The apparatus is linked to a personal computer where it can be set up through a host application.

Wack the Mole [23], a mixed digital/physical multiplayer game developed at the Umea University of Sweden in 2003 is another interesting way to use the Phidgets. The creators built a physical table with four holes where the 'mole' would pop up. The player has to use a

Fig. 8. The Antique Weather Clock: Smart devices, little effort. [21]

physical hammer and put it back down. Meanwhile another player on the computer could perform the same task in a digital way using his mouse. The software on the computer is aware of the table at all times, sensing a push down with the use of mechanical switches. A servo motor was used to release the moles. Additionally a motion sensor was included to attract potentional players that walk by the table by playing a jingle. 12 Phidgets were used to realize this playboard. A complete list plus wiring information can be found on the project website.

4.3 littleBits

Simulating basic functions is the focus for the littleBits example projects. What is shown on their website are fastly created mockups of usually highly complex devices: a smart phone with a touch screen and a button, a single button gaming device with a function to vibrate, a fake remote control. Some of the work can be seen in fig. 9. Functionalities are rather basic but the fact that the creation of the described mockups didn't take longer than 30 minutes each is noteable. No detailed information can be given as the toolkit isn't even released yet. The examples given should point out though that for rapid prototyping such very basic sets of modules seem to be the first choice.

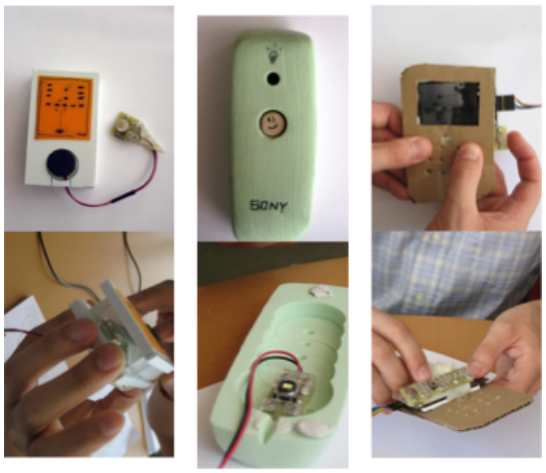


Fig. 9. littleBits Example Projects: Rapid Prototyping with little effort. [14]

5 Conclusion

Seven electrical hardware kits have been presented, analyzed and put into relation using their given parameters and available examples. To summarize and conclude this paper a few interesting observations can be put together.

Huge variety. No hardware kit is like the other. Although I just categorized some of the toolkits available and put them into certain groups every kit comes from its very own unique background, puts different hardware- and software architectures into focus and therefore attracts certain target groups. Interested people should take their time before investing in a kit as with a little bit of research online the best fit can be identified.

Online Communities. Web 2.0 communication plays a huge role with most of projects described. Forums, example code, example projects or FAQ sections are available, users are taking advantage of it. It is possible to use those aspects as another indicator on how well the kit fits with one's needs, how popular it is and most importantly, if quality and service are adequate and fair.

Software skills become the focus. None of the kits presented requires a deep understanding of hardware contexts. Elements can be put together very easily via cables, magnets or other basic mechanisms. Building electrical devices becomes playing Lego. The big obstacle is mainly to 'set the machine alive', to put digital content. The way the software part is organized within the toolkits varies a lot, there are high- and low-level solutions concerning micro controllers, host computers and programming environments. As this paper is written from a computer scientific point of view this point can be seen as very positive since it's the hardware obstacles that were identified as major barrier for tangible interfaces and ubiquitous devices or set ups.

No borders. When thinking about sketching or building devices (future) engineers should be aware that using a hardware toolkit doesn't mean they will be moving around in a self-contained system. Anything can be used to reach the goals and there will be very few cases where an apparatus is built out of pieces from the kit only. Also, anything that provides tools to build hardware devices can be seen as a hardware toolkit. This fact is well presented in this paper by covering both Lego Mindstorms as opposed to tools like BASIC Stamps or Arduino.

All in all it has the be clear that this categorisation is only one of many possible models. As software skills become the focus it would for example be plausible to put this subject into focus as well. Another approach would be to go into a specific detail of prototyping like speed, robustness or possibilities in variety. This might require actual studies though. In fact some of the frameworks can also be used to create actual devices, not only simulations: another parameter to distinguish the different kits.
The choice of toolkits could have been different, too. The goal was to cover a field that is representative, that shows the reader the many different alternatives there are. Again, for more information see [2] and [19] for more information on other toolkits and frameworks.

6 Future Work

As no concrete evaluation was performed to gather additional information about the value of the different toolkits only given data was used for this summary. It would be interesting to work on a more empirically driven comparison with specific tasks and regulations. Information like the time needed to finish a device or exact pricing comparisons would then be possible. A more detailed analysis of the software environments would be important and very helpful as information about these aspects of the toolkits is rather rare.

REFERENCES

[1] Brygg Ullmer and Hiroshi Ishii: Emerging Frameworks for Tangible User Interfaces. In Human-Computer Interaction in the New Millenium, John M. Carroll, ed.; Addison-Wesley, August 2001, pp. 579-601.

[2] Camille Moussette: Tangible interaction toolkits for designers. UmeaŁ Institute of Design, UmeaŁ University, UmeaŁ, SE-901 87, Sweden.

[3] Matthew Cottam and Katie Wray: Sketching Tangible Interfaces: Creating an Electronic Palette for the Design Community. Published by the IEEE Computer Society in IEEE Computer Graphics and Applications, 2009.

[4] Bjorn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, Jennifer Gee: Reflective Physical Prototyping through Integrated Design, Test, and Analysis. UIST06, October 1518, 2006, Montreux, Switzerland.

[5] Saul Greenberg and Michael Boyle: Customizable Physical Interfaces for Interacting with Conventional Applications. UIST02, October 27-30, 2002, Paris, FRANCE.

[6] Saul Greenberg and Chester Fitchett: Phidgets: Easy Development of Physical Interfaces through Physical Widgets. UIST 0l Orlando FI.A.

[7] Margot Brereton and Ben McGarry: An Observational Study of How Objects Support Engineering Design Thinking and Communication: Implications for the design of tangible media. CHI '2000 The Hague, Amsterdam.

[8] Snyder C (2003): Paper prototyping: the fast and easy way to design and refine user interfaces. Morgan Kaufmann Publishers, San Francisco.

[9] Nils Dahlbck, Arne Jnsson, Lars Ahrenberg: Wizard of Oz studies: why and how. Proceedings of the 1st international conference on Intelligent user interfaces, 1993, Orlando, Florida, United States.

[10] Joan M. Greenbaum, Morten Kyng: Cardboard Computers: Mocking-it-up or Hands-on the Future. Design at work: cooperative design of computer systems, Routledge, 1991.

[11] Ayah Bdeir: Electronics as material: littleBits. Proceedings of the Third International Conference on Tangible and Embedded Interaction (TEI'09), Feb 16-18 2009, Cambridge, UK.

[12] http://arduino.cc/

[13] http://www.parallax.com/tabid/295/Default.aspx

[14] http://www.littlebits.cc/index.html

[15] http://www.seeedstudio.com/depot/electronic-brick-c-48.html

[16] http://www.phidgets.com/

[17] http://mindstorms.lego.com/en-us/default.aspx

[18] http://www.buglabs.net/products

[19] http://www.partly-cloudy.com/misc/

[20] http://www.buglabs.net/applications

[21] http://www.thisispete.com/weatherclock/

[22] http://grouplab.cpsc.ucalgary.ca/Videos/2006-WatchDog-Friesen-1.02

[23] http://thieumsweb.free.fr/english/mole/

[24] http://beagleboard.org/

[25] http://www.mat.ucsb.edu/ dano/CUI/

[26] http://www.infusionsystems.com

[27] http://www.makingthings.com/

[28] http://www.nerdkits.com/kits/

[29] Saul Greenberg: Teaching human computer interaction to programmers. interactions, Volume 3 , Issue 4 (July/Aug. 1996).

All websites accessed on January 13th, 2010.

Prototyping in Physical Computing - Sketching in Software

Adalie Hemme

Abstract— The following paper introduces different software tools that help developers and designers to build prototypes for physical computing. It shows the importance of sketching in software and describes how different kind of people can use software toolkits and frameworks according to their needs. Most toolkits, mentioned in this paper, are based on visual programming, to reach highly diverse user groups from children, parents, all kinds of domain experts, designers, developers to regular programmers. They can be used to built mobile phones and all kind of information applications. The procedure, that most toolkits work with, is to connect physical hardware to a port in a software workbench, in which a duplicate of the pice of hardware is generated. These duplicate can be arranged in the same way the hardware is set up and not present hardware can mostly be simulated by the software. In the workbench the behavior of the devices can than be authored and controlled for the specific triggers on the hardware. Some toolkits offer complex tools that test and analyze workflows, so the prototypes can be adjusted to fit all the user needs. In my work I want to give a rough insight, to help understand these software tools, that mirror virtually physical computing prototypes and show the diversity of tools for the different fields in which physical computing is possible, and the diversity of users, that can profit by sketching prototypes for their needs.

Index Terms—Physical Computing, Visual Programming, Prototyping, Software toolkits and frameworks

1 Introduction

Prototyping is very important in all kind of development processes. In Phyisical Computing it might even be more important. Almost every object in our world can be part of a physical computing device, because it contains so many different fields. Therefore the development process is complex and costly and errors can make expensive hardware pieces unusable. Here Software is needed to sketch the entire functionality of the object of interest in a virtual workbench. This will help everybody who wants to build physical computing devices. By building a prototype designers and programmers can sketch their full creativity in form and functionality of objects. Software tookits make it easier and cheaper to evaluate the usability and functionality of devices and they minimize the time for devolpment. Another advantage of prototypes is that changes can be made much faster, by adjusting the functionality in the workbench, therefore development errors in the final product can be avoided[1].

Physical Computing means building interactive physical systems, software and hardware. Phsysical Systems integrate digital information with everyday physical objects [10]. Tangible Interfaces are used in Physical Computing, referring to added sensor technologies to electronic devices. Additionally any kind of sensory interface is accepted, like force sensors, accelerometers, and microphones. This physical input can control graphical or audio output in physical computing. Ubiquitous Computing is another part of physical computing. Integrating computer elements in a wide range of previously analog devices, enlarging the computer technologies to enter our everyday objects [14]. Examples are smart refrigerators, which know what food needs to be added, or chairs that know, who is sitting on it, measuring the weight and profile of the people announced to it. Technologies for Physical Computing are becoming more relevant in economics and therefore the tools, which help prototype new objects are getting more and more important [4].

Developing Physical Computing objects is spread upon very technical and hardware based fields. A prototyping tool should accomplish both: programming interactions and devices, designing highly usable interfaces and fitting the needs and knowledges of different user groups. Finding a tool that fulfils all these requirements is hard to develop and to find. These tools need to be designed such that both designers and programmers are enabled to work easily with them, without getting help from the other side, or having to read lots of instructions to use the tools. To enable people to interact with digital information by integrating it with everyday physical object is great. Programmers will have to think differently, they need to abstract the physical input, which is very different from user interface development [16].

Considering Graphical User Interfaces (GUIs) there are many tools, that help to prototype GUIs. About 20 years ago, when substantial raster-graphics expertise was required, there have not been many tools, helping to reduce the development time and ease users to work with the programs. Now there are already some tools, that enable programmers, who have few hardware experience, to work with physical input, as GUI toolkits have enabled programmers. who are not raster graphics experts. to build GUIs [16].

In this paper, some of the toolkits and frameworks that assist prototyping in Physical Computing are introduced. Focusing on software, that primarily supports visual programming, for the connected hardware. The software toolkits can be classified by the user groups they are made for and by the different kinds of domains, that can be build with them.

2 Overview of toolkits and frameworks

To build physical computing systems, which interact between humans and the digital world, tools are needed to help developers and designers to realize their ideas. Developers usually have a rough understanding of design issues, and designers can often not estimate, how hard it will be to implement their ideas. Therefore tools are required, to help developers to build applications and visual programmers to build application logics. That makes it easier to communicate with each other and implement changes on the product. Therefore toolkits have to respond to this existing gap, between designers and programmers, so that both sides can use the software and hardware tools, without having an detailed understanding of the foreign field [11].

To try out prototypes some hardware usually needs to be attached. This hardware should evaluate, if they work the way their behavior was authored to in the software. Examples for hardware are:

- Sensors, to enable systems and devices to perceive their environment and allow the reaction to the perceived facts by processing the gathered information.
- Actuators, which react to changes in the environment or in the context, to get the appropriate feedback in place. The are many different actuators from LEDs to robotic devices. It is important to choose the right actuator that fits the applications needs.

- *Adalie Hemme is studying Media Informatics at the University of Munich, Germany, E-mail: Adalie.Hemme@campus.lmu.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Prototyping, 2009/2010*

- Displaying Technology, like small and large sized displays, and projection technologies for public screens, and flexible energy saving screens for eBooks or wearable computing.

Existing sets of hardware to sketch prototypes with, are already available and widely used, like Arduino, Smart It's, the Calder toolkit, Phidgets, or Lego Mindstorms [13][2]. Many software frameworks already work with these hardware sets or have their own hardware set [7]. There are many ways to connect the mentioned hardware sets to the software toolkits. WLAN, Bluetooth, IrDA, GPRS, UMTS or ZIG-Bee are common technologies, next to USB or FireWire. The hardware sets used depends on the needs for the prototype and on the costs willing to spend [11].

The hardware pieces are connected by arranging the parts, that are needed for the prototype. The behavior and functionality of the single hardware pieces to each other and the environment, can then be programmed with code, or arranged using visual programming with an appropriate software. To try out the prototype with its sketched functionality, the hardware needs to be used the way it was designed for. With pressing buttons or using slides, the prototypes behavior can either be seen on the device, if it has visual or audio output, or in the software workbench on the computer.

As physical computing is becoming more and more economical, sketching with software is not only important to lower the costs and development time, it is also interesting for just sketching out individual ideas or new business ideas. With the prototypes in gereral the design and usability can be improved, due to the ability to fully test the prototype and adjust it to the best. With building more prototypes, the developers are getting more familiar with the toolkits, so the entire development process is becoming shorter, and costs can also be reduced this way. In addition there is less communication needed between the different employees, as they do not need to help each other that much, because the toolkits are enabling them to build prototypes, in fields they are not really familiar with. For example a designer can save time and money by putting the hardware together like this [2].

3 Analysis of the existing software for designers (visual programming) and programmers

In this section six different toolkits are introduced and their functionality is roughly explained. The frameworks are mainly aimed at users that can program a little bit or have a design background and which are familiar with a small technical understanding. They are made for different reasearch needs. The Papier Mâché is a special architecture that helps create software event abstractions, in which everyday objects are detected, by different input technologies as input of interest for designers. The framwork focuses on identity-based input and discrete events, to assist designers to rapidly exchange input technologies (see section 3.1) [6]. The d.tools framework is specified on devices like mp3 players, phones, and digital cameras (see section 3.2) and uses its own hardware components. The Quartz Composer (see section 3.4) is not a Physical Computing toolkit, but it is a good visual programming toolkit and reimplemented in some frameworks like the iStuff mobile. The iStuff mobile is for sketching with mobile phones (see section 3.5). And the NETLab toolkit is designed for students and designers, to sketch with sensors to author motors and projections (see section 3.6).

3.1 Papier Mâché

Papier Mâché is an open-source Java toolkit written using the Java Media Framework and Advanced imaging API's. It is made to abstract a collection of input from everyday objects which are tracked by a camera, tagged with barcodes, or have RFID tags. The developer has two tasks prototyping with Papier Mâché, declaring the input he is interested in and mapping this input to application behavior. With Papier Mâché a developer can easily implement an object with one technology and later retarget the object to another technology [16].

A developer first has to select input types, like RFID or vision, which gets acquired and interpreted in the input layer and generates a physical object out of it. Developers do not have to deal with the connection of input devices or with events generated by them. The hardware connected to the computer that the developer will be working with, needs to implement the Input Device marker interface. This is a design pattern, with which an interface is created without any methods. So each input device gets its own API to deal with the underlying hardware. Hence, the implementing class is responsible for providing input acquisition. There are specific classes, for the different kinds of hardware devices, implementing the marker interface with the API for the Papier Mâché toolkit support. Special functionality can be added for specific hardware, which inherits its major behavior from the general hardware type. Hardware input behavior can although be simulated, when it is not available through the FilesImageInput class. The PhobProducer (Physical Object Producer) generates events, once a input source is selected, representing the adding, updating and removal of an object., which he obtains in the sensor view. All technologies produce the same events by holding different information about the objects. RFID has the tag and the reader IDs, where Vision has the size, colour, location, orientation, and the bounding box of the object, which applications can use to determine the application behavior. When a RFID tag is placed in the range of a user a PhobAdded event occurs, if the reader does not report an objects presence, it is detected as removed. Vision Events are interpreted by camera calibration, image segmentation, and event creation and dispatching, while using edge detection or background and object segmentation [16].

There are three levels of handling behavior of detected objects

- PhobEvent instances carry information about the objects, being new physical input, detected by a PhobProducer.
- AssociationFactory instances, which are interfaces to create Association Elts, provide mechanisms for creating and modifying application logics.
- BindingManager, it receives all PhobEvents automatically and uses the AssociationFactory to create AssociationElts. It although manages the flow of events and creates behaviors. In addition it contains a map data structure with past and present bindings, when a new event is detected he compares it with the classifier, which invokes application behavior for matching elements.

The developers goal is to instantiate parameterized classifiers and behaviors, to build applications, which can often be created by parameterize existing library classes. More complex functionality can be implemented by building an own application with custom behavior. The library includes some media manipulation actions like pause, rewind, and fast forward, which can be used on media clips for the behavior of applications. The library has three classes of physical devices. First the electronic tags, like RFID. Second the barcodes, which must be manually tagged before they can be used. With this way any object can be appropriate for use. The third class is image analysis, it is much more flexible, because it can simply use an object as an image or it can be used for pure capture or pure recognition [16].

The application developer has a monitoring window shown as a three-level-tree (see figure 1), which shows the current state of the system. The first level shows the PhobProducer types, like RFID, barcode or vision. The next level shows the instances of objects creating input events from the PhobProducer, and the third level shows the currently visible objects. Papier Mâché offers Wizard of Oz (WOz) control to use not available input for the PhobProducer, where events can be created as if they came from a sensor. Furthermore provides the graphical interface from Papier Mâché authoring information, next to the monitoring information. Working with the authoring facilities, being realized by xml, allows the developer to create and modify the runtime code of the applications. For adding a new classifier for a new behavior, the input technology needs to be selected, to create a new binding between a class of physical input and an application behavior. Hence, for custom behavior developers can implement their own AssociationElts and then specify only the input parameters they are interested in, like "filtering" objects by a specific shape, colour, or size [16].

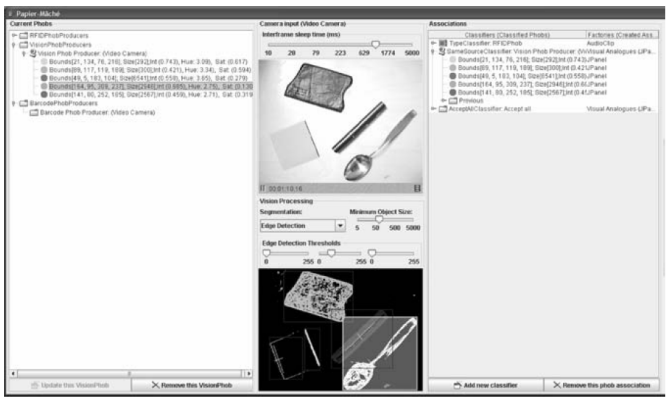


Fig. 1. Papier Mâché Monitoring window [16]

3.2 D.tools

D.tools is a design tool, build by the University of Stanford, and can be installed as a full Eclipse plug-in. Its an iterative-design-centred approach to prototyping physical user interfaces for mobile phones, digital cameras, music players and similar devices. In Addition to physical programming the toolkit is usind textual programming to provide a higher ceiling. D.tools has a hardware interface, which is extensible to prior system libraries. It offers three points to extend hardware to. The hardware to Computer interface, the intrahardware communication level, and the circuit level. Integrating design, test, and analysis of information appliances makes it easier to evaluate the prototypes. While a user is running a test all events are shown in the workbench and on the hardware, d.tools logs all device events and state transitions on video and automatically classifies these hardware events and shows the analogue state chart to the hardware events. Single user or multiple user videos can be analysis at the same time, where the logs from the videos can be compared, a interaction patterns get visible and a history about all the transitions is displayed. The logs of multiple users can be compared in video matrix, where the rows correspond to the users and the columns show categories like hardware events or comprise of states [7].

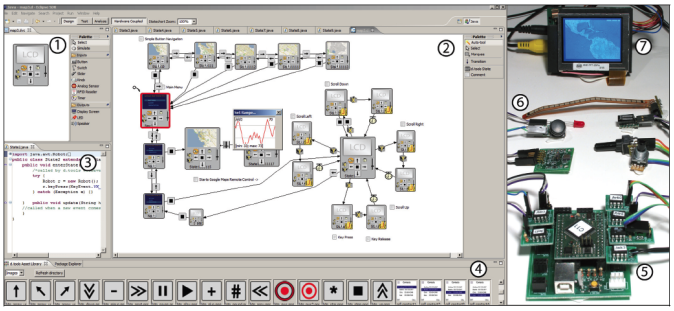


Fig. 2. d.tools workbench with connected hardware [7]

D.tools is rather made for designers, who can place physical controller, sensors and output devices directly on their prototype. The library includes many input and output technologies. The software representation of the I/O components can be graphically arranged in the design mode. The behavior of these components can be graphically authored and then being tested on the physical devise, which is always connected to the visual interaction model. D.tools includes a framework to visually program hardware behavior. The hardware is connected to a d.tools hardware interface, which is the first step in the design process. The connected devices announce themselves automatically to the framework, where graphical duals are created of themselves. Hardware component that are not available, can be included as a visual-only input and act as a Wizard of Oz during the test time, or the designer can try to manipulate the hardware or they can try to imitate hardware events by using a simulation tool. Designers can create, reform, and arrange input and output devices in the device editor. In Addition they are able to create their appearance by using a large library of buttons, slides, LED's, and all kinds of components. Interaction graphs are build to create the prototypes behavior (see figure 2). They have to describe the content for the outputs, including screen images, sounds, LED behaviors for all the different states. The control flow of an application is represented by transitions, they are represented as arrows between two states in the workbench and define the rules for a change in the states, while some hardware event occurs or a timer is running out. The visual workflows are somehow self-explaining, but with increasing functionality text notes are useful and can be placed everywhere on the state charts. Instead of numbering the range of sensors and other hardware devices, the designer can run a real time simulation for the device of interest and set the upper and lower threshold while for example moving a slider [7].

For very complex prototypes d.tools offers parallel state charts, to give one button different functions depending on the priorities or context. Furthermore it is extending state charts with writing Java code to visual states to specify their behavior. For this d.tools uses eclipse to help writing correct code [7].

3.3 Exemplar

Exemplar was although developed by the University of Stanford and works as a full Eclipse plug-in and is able to communicate with the d.tools hardware interface, Wiring and Arduino. Its helps prototyping rapidly sensor based interactions. Exemplar offers new techniques to author sensor based interactions through programming by demonstration using a graphical direct manipulation interface. As sensors can be used accelerometers, bend sensors, IR rangers, light sensors, touch sensors and sensitive resistors. Data arrives on the Graphical User Interface (GUI) on the left side of the screen and can be manipulated to discrete or continuous events on the right sight of the screen. All sensor data is shown live and can be uniquely filtered and then be send to other applications or being converted into discrete events. The event definitions for each active sensor are shown in the GUI and marked with individual colours [3].

3.4 Quartz Composer

Apple produced the Quartz Composer, which is a visual programming tool provided as part of the integrated development environment for Mac OS X's XCode. The tool is made to enable users to build graphics processing modules without writing any code. Its original function was to be a programming environment for computer graphics, 3D graphics and animation graphics. The Quartz Composer makes the developers work with patches, which are base processing units that execute and produce results. On the patches are circles, which are ports representing input and output, and are passing data through the patches. Costum patches can be written with JavaScript to modify and convert input events. The complexity of the program is entirely hidden [9].

Quartz Composer is able to create editor compositions, which are procedural motion graphics programs created by assembling patches in a workflow for data processing and rendering (see figure 3). Compositions have input parameters and create output, they can be put into any other workflow. There are three kinds of patches with different colored title bars:

- consumer patches, which render results to a destination.
- processor patches, which process data at defined interval, or if input values are changed.
- provider patches, which provide data from an outside source to a composition.

Patches can be modified to custom needs or new patches can be programmed by code or with visual programming. There are also macro patches, which can use or call other macros. Macros need sub patches,

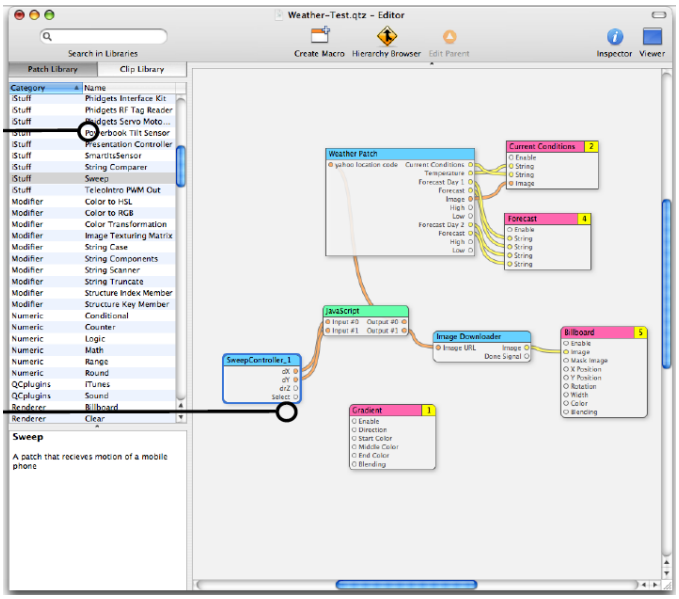


Fig. 3. Quartz Composer patches with library [2]

to create objects to work with. For this Quartz Composer offers an evaluation path, where it is graphically shown when and how often each patch, in different levels, in a composition executes and which macro patch is leading to another macro patch. With making custom macros, patch collections and their connections can be packaged to one macro, to reduce complexity [8].

3.5 iStuff Mobile

The iStuff Mobile framework helps building prototypes for mobile phones while using existing phones. Due mobile phones in general are not build for research needs and it is hard to extend the hardware of an mobile phone, as it is packaged very small for the commercial needs. iStuff mobile allows to add external hardware to the phones with using Smart-Its sensor network modules. These new composed hardware pieces can than communicate with an visual programming environment. To do so a background cell phone application that allows prototyping with any foreground application, a reversion of Apples Quartz Composer is needed. In Addition it is necessary to have a sensor network proxy for configuring sensors without changing their software. For this iStuff Mobile reused and adapted some existing and proven software as iStuff. iStuff offers patch panel with which messages can be intercepted and be rewritten, so that incompatible components can communicate over a network. The mappings of inputs and outputs can be specified during the runtime, the Event Heap and the Quartz Composer. iStuff is offering the feature, that allows to try out interactions, current mobile phones are not able to. Therefore iStuff Mobile can be used as software parts are distributed between different computers. Hence the communication takes place over the Event Heap, without having an explicit connection between the components. Smart Its and the mobile phone are not supposed to communicate over the Event Heap, so they use a proxy strategies, where an external process can send and receive directly with the hardware [2].

The mobile phone application is split into two parts. The part visible to the user is the application the user interacts with. This uses Symbian Series 60 operating system in the background. It runs the new designed application from the designer, where events can be send to the mobile phone proxy via Bluetooth (see figure 4). The background supports Bluetooth, sound playback, vibrator control, key capture, profile control, camera control and many others. It then sends the events to the foreground application, which can be exchanged with any other foreground application. Existing applications on the foreground can be taken over and new applications can be programmed for the phone using Java [2].

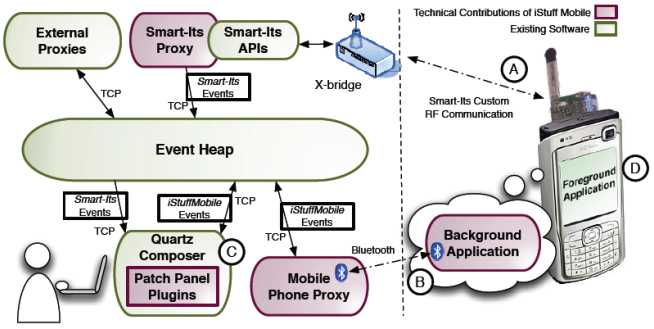


Fig. 4. iStuff Mobile architecture [2]

Smart Its were chosen for the sensor network support, which provide a remote procedure call interface. Therefore sensors can be reconfigured without changing the code on the sensor boards. The GUI was designed in a way, that the user can configure and activate all different sensors, for light, microphone, voltage and other sensors, concerning one particular scenario, at a time. Any other Network platform can be used for the iStuff Mobile just by implementing a new Event Heap proxy and a new plug-in for the Visual Programming platform, to control the information flow in the GUI [2].

For Visual Programming iStuff Mobile is using a configured Quartz Composer, where all changes are immediately added to the application during run-time. Each iStuff proxy and new data processing modules were added to the Quartz Composer library [2].

Some examples for new applications, which were prototyped with the iStuff Mobile, are [2]:

- If a call or message arrives in the sound mode, the phone will just vibrate, if a user is holding his phone in its hand, it is just vibrating instead of ringing when a call or message arrives. This is because of the pressure sensor is used, which detects whether the phone is hold in the hand or in a pocked and not just laying somewhere.
- Using Quartz Composer together with Java Script enables iStuff Mobile to type texts without using the keys but just tilting the phone in one of four directions by which different letters or symbols can be chosen.
- Building an application for interactive displays. This is very easy and works accurate over te camera capture point.
- Controlling different displays with a mobile phone. Thus the presentation shows the old slide in one screen and the new slide in another. This was realized using proxys running on different machines, that are listening to a particular event name.

3.6 NETLab Toolkit

The New Ecology of Things Lab Toolkit (NETLab toolkit) is an open-source set of software tool to help designers or students to sketch in hardware in addition to create interactive objects and spaces, without having to program. Furthermore it can be used to augment the toolkit, the development environment is similar to Flash. The Toolkit consists of three parts:

- Flash Widgets, which enables the drag-and-drop interface. It establishes the communication to external hardware by placing the right components on the Flash stage and setting some options, which make the extern hardware interact with the Flash.
- the Hub, which combines all hardware connections brings them to and from the Flash environment. It connects the hardware through a socket interface to the computer and handles the communication tasks. The connection can be made by Make Controller, Arduino, XBee and OSCs, like the Wii controller.

- the Media Control, which works with specialized media like LED systems, multi-channel audio with up to eight sound channels and DMX lightning systems.

There are a lot of sensors, like wireless sensors, Wii Remote input, controls motors and LEDs, which communicate with MIDI devices, control sound, graphics, videos in Flash, and the DMX computer controlled lighting equipment. The last mentioned can be used by designers to program with drag-and-drop, which enables them to control video projections or motors by using the connected sensors. Prototyping this fast an easily with hardware components, which is a unfamiliar field to designers, enables them to focus on the design process. This way they do not have to communicate with technical persons to try out their new ideas [17].

4 Toolkits for Children and Layman

In this section two toolkits are introduced, which help children and non-technical users to build physical computing devices. These toolkits are build to ease users without any technical background or interest to build individual applications with an educational interest. Both toolkits also offer a different level of programming, for which more programming knowledge is needed, to build more complex applications or to enlarge the programming options for non technical users. Edu Wear is primarily made for children. With the toolkit they can add funtionality to their clothing and lern about technology (see section 4.1). The ESPranto SDK addresses different user groups, designed with a educational interest, so that non programmers can slowly become programmers, and offers a working access at all the steps fitting the users knowledge (see section 4.2).

4.1 EduWear

EduWear is made for children to teach them constructionist studying with making hidden technologies visible. EduWear takes cloth with added sensors and actuators as the toolkit to be scratched with. By taking fashion as the toolkit for augmenting the functionality of clothing. Therefore it is easier to get the childrens interest to experiment with. It consists of a programming languages, microcontrollers, actuators, sensors and connections between the cloth and the technologies (see figure 5). Arduino is used in EduWear, the board layout is split up in three parts. The "programmers board" which is connected to the computer while programming the microcontroller. The "power board" to load the "main board" with the needed voltage. The "main board" is placed within the cloth. Instead of using "Arduino" programming language, because it is too difficult for children, "Amici" is used, which works with dragging and dropping icons to program visually. The programmed graphic blocks are generated into Arduino code, to make children get used to the textual programming language [15].

Fig. 5. EduWear software screenshot and the textile databus with actuators and sensors [15]

Besides tilt-switches, light sensors and LED's as actuators, EduWear is using textile technologies. A stretch sensitive canvas which changes the resistance while stretching the conductible yarn. And textile switches, where two conductible layer are isolated by another layer and get connected by pressure. A Data bus connects the actuators and sensors with the microcontroller. It uses conductible yarn to the buttons witch with the sensors are fitted to the clothing [15].

4.2 ESPranto SDK

The ESPranto Software Development Kit is a sensor and actuator based application designed for non-technical users, like teachers or psychologists, to build educational toys, games and interactive lightning systems. These users usually have very specific ideas, which they can implement with a given set of tools to build simple applications. With increasing programming skills, that is achieved by always give the user a feedback about the currently used tools. Users are enabled to build more and more complex applications with more flexibility in the choice of tools. The SDK although allows all other kinds of users to build applications. Non-technical users like parents or teachers are usually good content creators and can develop simple applications and professional content with the toolkit. Domain experts like psychologists, game designers or therapists can build very specified applications according to their domain knowledge easily. Technical experts like programmers may work with the other two user groups to construct the building blocks for their applications and build new building blocks in general for themselves. The forth user group are legacy programmers that want to use the ESP runtime environment to couple with other systems not using the ESPranto. Error messages are always matching the frame of reference for the user This is why a non-technical person, gets non error messages, he just sees whether his program compiles or not [18].

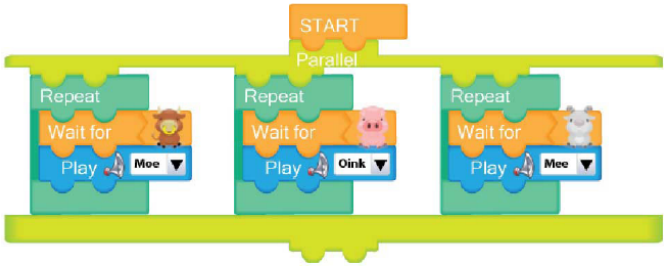


Fig. 6. A program in the graphical layer to simulate animal sounds [18]

The ESPranto SDK is part of the Edutainment Sensor Platform (ESP). The platform was developed to allow users to combine different types of sensors and actuators and create for these different hardware combinations software. ESPranto SDK runs on a computer to build software for the applications. The ESP runtime environment runs on an embedded processor, which is placed into toys, and executes the compiled code by the ESPranto SDK. The ESP runtime environment consists of:

- drivers for the sensors and actuators. These drivers are preinstalled for the currently used hardware.
- the ESPranto byte code interpreter.
- an input abstraction layer. It feeds the input events from the sensor drivers to the byte code interpreter.
- an output abstraction layer. It feeds the output events from the byte code interpreter to the actuator drivers.

Due to the different types of users with different goals, needs and skills the SDK is split up into four layers. The layers enhance a different level of flexibility and complexity. Applications can be made in one layers output also they work for the layer below [18].

The graphical editing layer works by dragging and dropping puzzle pieces, of which each is building a building block, to create a program (see figure 6). The pieces are connected vertically to form sequences, some although form horizontal connection to allow parametric sing. This layer offers the least flexibility and is made for layman. Errors are prevented by only using correctly programmed puzzle pieces. By using ESPranto language so no semantic errors are can be created by connecting them. Depending on the connected hardware only a correct set of tools is offered to avoid errors [18].

The next layer is the macro layer. A macro can be understand as the textual counterpart to a puzzle piece, which is interpreted into macro before it compiles. This layer is made for domain expert that needs more flexibility. The programming language in this layer is very easy and can be bound very closely to a specific hardware configuration. Therefor a professional programmer builds a library of macros with the basic software building blocks. With macros statements can be passed as parameters to ease the functionality, error messages are given before the macros are expanded, and automatic type inference [18].

The third layer is the ESPranto kernel layer. It consists of a small set of basic behaviour the kernel statements, which are the same, as in the programming language Esterel [5]. When an application compiles from the layers above the code might be expanded into kernel statements, which the compiler than expresses into byte code. The Marco and the kernel layer are not strictly separated, both domain experts and programmers can combine and use both layers [18].

The last layer is for legacy layer. It emerged before the SDK existed, when the programmers build the applications using C++ and Java, to communicate with the ESP runtime environment. This layer is useful, when applications have to be integrated into existing software, made with different programming languages, by making just few changes to the interface [18].

5 Outlook to the Future

As there are many tools for different user groups, concerning on different development fields and kinds, some with commercial ambitions, others with educational ambitions, they are all designed to fit the specific needs. Most of them using visual programming, to ease the programming process using self-explaining icons, to lower the customization with the programs. This might be easier for different user groups like [12]:

- children, to make them learn programming in a playful way, like EduWear focused on.
- first-time programmers, which can learn the structures and basis concepts of programming over the graphical construction.
- End user, which can try to personalize their programs, or build programs on their own.
- domain experts, who do not need programming skill, but need to built programs for their studies.
- programmers, who want to try out new approaches and build fast and error free programs quick.

Especially the ESPranto SDK was build, to fit many of the user groups, mentioned above. With having the basic idea of more and more non-programmers are getting involved in building content, and enabling then with gained experience to build more complex programs using less graphical programming. But they still want to improve the error messages, making them more understandable for starters, and to enlarge their area of application to more sensor and actuator based applications [18].

Compared to that, d.tools rather concentrates on building devices that could be economically manufactured, so they put a higher focus on integrating support for testing and analyzing the workflow of prototypes. In the future they want to be able working without an connection to the computer, by connecting components to an embedded Intel XScale platform, which can execute interaction models and to enable on-board graphics wither higher than 8 bit microcontrollers [7].

iStuff Mobile focuses on integrating the mobile phone more in ubiquitous computing scenarios, hoping to advance the pace of innovation and improve the quality of interface designs in ubiquitous computing [2].

All of the mentioned tools, are planning on improving their toolkits, by reducing errors, enabling them to be more portable to other technologies, or want to become easier to use to their desired user group. As all the tools are free to work with or even open-source, finding this easier to have more operators, helping them to improve the products. Many of the toolkits come from universities or have a research background. Here it makes sense to keep the access open to everybody interested in physical computing, as there are so many opportunities to develop very diverse products, so the more people are enabled to work with the toolkits, the faster research results can be successful. As the field for new innovations in physical computing is so large, it might be hard to find one toolkit, which can prototype any application. Therefore it could be more likely that some standard prototyping toolkits for different fields of applications might arise, instead of one universal large tool. This would not even be useful, as physical computing can be used by so many different user groups, with different interests and a different educational backgrounds, and all the results are of node from a economical and research view.

6 Conclusion

After finding out, that sketching in software for physical computing is becoming more important not only for the economy the paper shows some different toolkits and frameworks, helping to sketch prototype primarily by visual programming. I found that prototyping is a good way to enable non-technical users and young people to sketch with physical computing. Nevertheless more advanced people can still work with the visuall tools and add complexity with programming code. I like the way of physical programming because it closes the gap between programmers and designers, who work on research or economical products. Most of the toolkits are free, even the one apple produces, which seems to be important for fundamental research. This way as many people as possible have the opportunity to try out the frameworks and help improving them. All the toolkits enabling a wide potential user group to transform many individual ideas, which can lead to great new projects and might lead to products that enrich our lives. Reading the papers made me want to sketch with the toolkits, too. In Fact some frameworks, like d.tools, were already mentioned in many other papers. A great advantage is , that the introduced toolkits work with similar principles, but for different user groups and application areas.In this way interested people can easily switch to a different toolkit, when they are interested in a different development field, like from building child applications, to move over to build cell phone applications. But it would be more useful, if the toolkits would work with the same programming languages, for the more advanced users and not only use the same visual programming paradigm. In addition a very helpful concept is ESPeranto SDK. It has different levels of entry, for different kinds of users, to enable a learning progress for the users and give everybody an access.

In general I would suggest to have two major toolkits for different research fields. One for children and non-technical persons to build less complex applications and primarily focusing on bringing these people in touch with technologies at all. And one for very commercial needs, which ease developers, to prototype mobile phones, music players, cameras and all kinds of everyday objects, which is super complex and extremely flexible and portable to all kinds of technologies and including all kind of technologies. Therefore developers would have no need to get to use a new toolkit again, just for building different kind of applications. Another useful toolkit would be one that enables normal people to build applications at home with the hardware already available in the house. To which connecting cheap sensors or actuators should be made very easy, so that everybody can add some customized physical computing to their environment. In the future physical computing will be more and more involved in our lives.Besides great projects for toolkits, which provide important fundamental research for the future, there will probably be many more new toolkits, focusing on different application areas, helping to let our physical world interact with the computers.

References

[1] J. A. J. Andrew Sears. *Human-Computer Interaction: Development Process*. CRC Press, 2009.

[2] R. Ballagas, F. Memon, R. Reiners, and J. Borchers. istuff mobile: rapidly prototyping new mobile phone interfaces for ubiquitous computing. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1107–1116, New York, NY, USA, 2007. ACM.

[3] S. K. M. M. Björn Hartman, Leith Abdulla. Exemplar, 12 2009.

[4] M. Cottam and K. Wray. Sketching tangible interfaces: Creating an electronic palette for the design community. *IEEE Computer Graphics and Applications*, 29(3):90–95, 2009.

[5] G. B. D. Potop-Butucaru, S.A. Edwards. *Compiling Esterel*. Springer, 2007.

[6] B. Hartmann, L. Abdulla, M. Mittal, and S. R. Klemmer. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In M. B. Rosson and D. J. Gilmore, editors, *CHI*, pages 145–154. ACM, 2007.

[7] B. Hartmann, S. R. Klemmer, M. Bernstein, L. Abdulla, B. Burr, A. Robinson-Mosher, and J. Gee. Reflective physical prototyping through integrated design, test, and analysis. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 299–308, New York, NY, USA, 2006. ACM.

[8] A. INC. Quartz composer user guide, 12 2009.

[9] H. Kimura, Y. Okuda, and T. Nakajima. Cookieflavors: Rapid composition framework for tangible media. In *NGMAST '07: Proceedings of the The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies*, pages 100–109, Washington, DC, USA, 2007. IEEE Computer Society.

[10] S. R. Klemmer, J. Li, J. Lin, and J. A. Landay. Papier-mache: toolkit support for tangible input. In E. Dykstra-Erickson and M. Tscheligi, editors, *CHI*, pages 399–406. ACM, 2004.

[11] A. S. Matthias Kranz. Prototyping smart objects for ubiquitous computing. Technical report, LMU Munich, 2005.

[12] P. Merkel. Visual programming im überblick. Technical report, Universiät Ulm, 04 2009.

[13] C. Moussette. Tangible interaction toolkits for designers. In *Scandinavian Student Interaction Design Research Conference*, 2007.

[14] B. Myers, S. E. Hudson, and R. Pausch. Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact.*, 7(1):3–28, 2000.

[15] M. Reichel. Eduwear: Ein construction kit für smarte textilien und wearable computing. In R. Koschke, O. Herzog, K.-H. Rödiger, and M. Ronthaler, editors, *GI Jahrestagung (1)*, volume 109 of *LNI*, pages 540–544. GI, 2007.

[16] J. A. L. Scott R. Klemmer. Toolkit support for integrating physical and digital interactions. In *Human-Computer Interaction*, volume 24, pages 315–366, 2009.

[17] P. van Allen. Netlab toolkit, 12 2009.

[18] R. van Herk, J. Verhaegh, and W. Fontijn. Espranto sdk: an adaptive programming environment for tangible applications. In D. R. O. Jr., R. B. Arthur, K. Hinckley, M. R. Morris, S. E. Hudson, and S. Greenberg, editors, *CHI*, pages 849–858. ACM, 2009.