
# Prototyping of User Interfaces for Mobile Applications

Springer

# T-Labs Series in Telecommunication Services

More information about this series at http://www.springer.com/series/10013

Benjamin Bähr

# Prototyping of User Interfaces for Mobile Applications

Springer

Benjamin Bähr
Quality and Usability Lab (TEL 18)
Technische Universität Berlin
Berlin
Germany

ISSN 2192-2810 ISSN 2192-2829 (electronic)
T-Labs Series in Telecommunication Services
ISBN 978-3-319-53209-7 ISBN 978-3-319-53210-3 (eBook)
DOI 10.1007/978-3-319-53210-3

Library of Congress Control Number: 2016963744



Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Acknowledgements

This text presents the results of the research I did for my dissertation. A number of persons supported me during this work. They shaped me in scientific debates, kept an open ear to my questions, or strengthened my back when work was tough. They helped to turn the past years into a time that was instructive, rich of experience, and overall very enjoyable to me. I do thank them very much.

First of all, there is Sebastian Möller. The commitment with which he supported me, even when his time was scarce, was exceptional and more I could ask for. He consulted me in countless meetings, kept replying to my needs in no time, and he established an atmosphere in his chair that was truly inspiring to me with its diversity and open-mindedness.

I am truly thankful for the support I received from Michael Rohs. Along with Sven Kratz, he put my scientific career onto track. Even though Michael left Berlin early to follow his academic career, he continued to support my research with regular remote mentoring and numerous meetings in Hannover.

Not less thankful I am for the support of Johann Habakuk Israel. With his constructive feedback he gave me inspiration and motivation even before I was able to gain him as a third supervisor. My work gained hugely from his input about the theory and psychology of sketching.

Further, I want to thank all the colleagues of my chair, the Quality and Usability Labs. They always showed interest and supported me in my work. I enjoyed an atmosphere, characterized not by concurrency, but by collegiality and helpfulness. Needless to say, I am truly happy about much friendship that evolved from my time at the chair.

Explicitly, I want to thank Irene Hube-Achter and Yasmin Hillebrenner for the administrative support and for steering my projects through the cliffs of bureaucracy!

Moreover, I want to thank my colleagues of the Telekom Innovation Laboratories that provided me with a generous and inspiring working surrounding.

I want to thank my fellow researchers of the Rethinking Prototyping research project that opened my view on the topic of prototyping to a more versatile and interdisciplinary perspective.

Thanks to my fellow stipendiats of the Prometei Graduate School that helped me getting into my research topic in the first years of my dissertation.

I want to thank the institutions that helped me in funding my research project, which are the DFG, the Einstein Stiftung Berlin, and the BMWi funded Software campus research grant.

Special thanks to Franzi and Rainer of the Berliner Ensemble, for helping me out with welding and serious metalwork in hard-hardware setup of my tabletop environment.

I am very thankful to all my friends and my family. They make my life!

I want to thank Alice for accompanying me throughout my dissertation, and turning this period into a wonderful part of my life.

# Contents

# Abbreviations

| | |
|---|---|
| ANOVA | Analysis of variances |
| ATT | Measure of attractiveness |
| AX | Axure prototyping software |
| BP | Blended prototyping |
| CAQ | Creativity assessment questionnaire |
| CSCW | Computer supported collaborated work |
| CSV | File format of comma separated values |
| CV | Computer vision |
| DR | Design reviews |
| DSLR | Digital single-lens reflex camera |
| GPU | Graphics processing unit |
| GUI | Graphical user interface |
| HCI | Human computer interaction |
| HQ-I | Measure of hedonic quality—identification |
| HQ-S | Measure of hedonic quality—stimulation |
| HTTPS | Hypertext transfer protocol secure |
| ICC | Intra-class correlation coefficient |
| JSON | Javascript object notation |
| JVM | Java virtual machine |
| MVC | Model view controller design pattern |
| OTAS | Observational teamwork assessment survey |
| PBP | Paper-based prototyping |
| PQ | Measure of pragmatic quality |
| PT | Prototype |
| RQ | Research question |
| SD | Standard deviation |
| SDK | Standard development kit |

| | |
|---|---|
| SE | Standard error |
| UI | User interface |
| URL | Uniform resource locator |
| XML | Extensible markup language |

# Chapter 1
# Introduction

Since the release of the first iPhone in 2007, mobile devices like smartphones and tablets are booming. In 2014 sales of smartphones worldwide topped 1.2 billion units, which was almost 30% higher than 2013.[1] This means that in average about every 6th human being on the planet bought a new device in the last year.

Smartphones give their users the opportunity to install additional software with apps, which can be comfortably downloaded from centralized app markets. The most popular of these markets, the iTunes Store that provides apps for Apple's iOS devices, was introduced in 2008. Seven years later, by June 25th 2015, the total number of app downloads from the iTunes Store alone passed the number 100 billion.[2]

Without a doubt, app development rapidly became a relevant field in the business of software development. In comparison to software for the mobile use, desktop software tends to offer holistic solutions in big software packages. Mobiles apps are mostly very small pieces of software that are targeted to solve a specific task. The main reason for this lies in different usage of both types of software: desktop software is typically used for a longer period of time to fulfill complex work, whereas mobile software is mostly used in between the daily routines as a service or to quickly gather a certain piece of information.

For the reason of mobile apps being comparably short in their range of functionality, they are usually developed in rather small teams, most often employed by smaller companies. Due to a shortage of a number of resources, especially in time and money, these groups work under a lot of pressure to come up with the *killer app* idea: the app providing the service everyone is looking for, in a spectacular design, and at exactly the right time. Bright shine the shooting-star-success-stories of the

[1] http://www.usatoday.com/story/tech/2015/03/03/apple-samsung-smartphones/24320385/ (last accessed 11th April 2016).
[2] http://www.theverge.com/2015/6/8/8739611/apple-wwdc-2015-stats-update (last accessed 11th April 2016).



B. Bähr, *Prototyping of User Interfaces for Mobile Applications*,
T-Labs Series in Telecommunication Services,
DOI 10.1007/978-3-319-53210-3_1

Instagrams, Evernotes, or WhatsApps in the sky. However, most app development projects fail. For small startup companies this means a failure of the whole business.

To develop high quality apps it is necessary to test them with users as early and as often as possible. Of key importance in this context is the user-interface, which builds the center of interaction between the application and its user. With tests, design decisions can be evaluated and wrong decisions become apparent at a stage early enough to be still corrected in the further development. As Szekely [145] melts it down in a concise advice: "Build an initial version of the interface, and then test it with users and revise it as many times as you have money and time for" (p. 76).

The quote in the previous paragraph originates from 1994. This underlines that the necessity to conduct user tests in software development projects is not new. In fact, it is much older than the history of mobile app development itself. The previous two decades of experience with graphical user interface design provides a new generation of mobile app developers with a profound knowledge base on processes and tools for a user driven software development.

One of such process models is the Usability Engineering Lifecycle, introduced by Nielsen [107]. The concept propagates a product development in form of an iterative refinement, driven by repeated user tests of design ideas with prototypes.

The Usability Engineering Lifecycle, as displayed in Fig. 1.1, is divided into five phases that are processed sequentially. At the end of the fifth stage, the process is repeated in a new iteration.

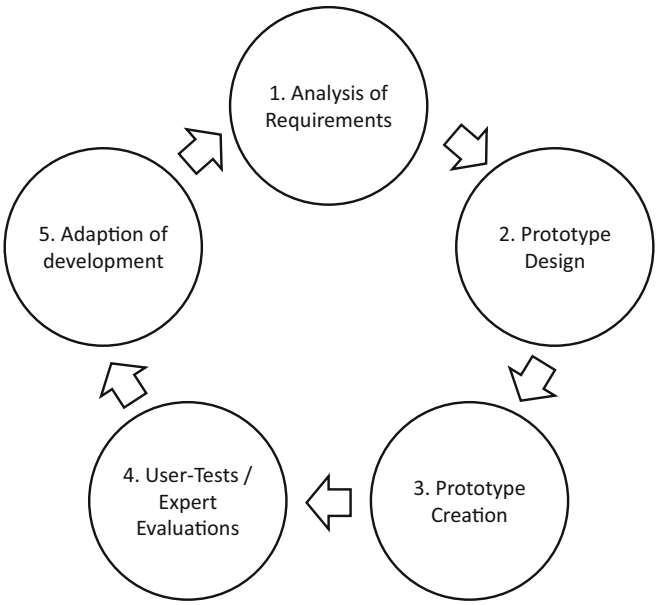


**Fig. 1.1** Usability engineering lifecycle

A cycle is initiated with the requirements analysis of the current product state, and the planning of the iteration cycle lying ahead. Followed by that, a prototype is designed that addresses the identified requirements, in other words, asks the question about the current product state that are currently most relevant. Now, the prototype is created and prepared for the testing. Tests can be done in form of expert evaluations, where domain experts provide feedback about the interface implementation with specific heuristics. Or the prototype is evaluated in a user-test, where test users are asked to try out the prototype to generate subjective or objective usage feedback. The generated feedback is now analyzed and consequences for the further development are drawn and implemented in the prototype. In the next step, a new iteration is started and the process is repeated for a newly developed prototype.

The Usability Engineering Lifecycle underlines the importance of prototypes for the user-centered design of software products. However, its applicability depends on the existence of prototyping tools that allow for a quick design and implementation of prototypes. Without tools that allow the implementation of prototypes with a reasonable low effort, the Usability Engineering Lifecycle cannot be sufficiently applied.

In later product phases, prototypes are usually created with the same tools that are also used to develop the final product version. However, for earlier design stages, these tools are too complex and slow. Here principle design decisions are made, which makes it highly advisable to produce quick prototypes to validate those ideas. This need is addressed by prototyping tools, which are designed to ease and accelerate the design and creation of testable prototypes.

Especially in early design stages, where principle design decisions are made, prototyping tools should be able to integrate all relevant persons into the prototyping process. This includes interdisciplinary teams of designers, programmers, project managers, and others. For the field of designing conventional software user interfaces for the stationary use, prototyping approaches like the paper-based prototyping are well established, which can be applied in earliest ideation phases already to leverage a creative group work on an evolving interface idea. As e.g. Sá et al. point out in [127] and [125], the transformation of such concepts to a prototyping tool that addresses the specific challenges of mobile app development is not yet sufficiently solved.

This research aims to gain a better understanding about how prototyping systems can improve the development of mobile applications. It aims to describe, which aspects of prototyping tools are most relevant for practitioners, and how such aspects might change in different development stages. Furthermore, it identifies a demand for improved prototyping systems in early development stages. It discusses a new approach that was conceived, implemented and tested by the author: The approach of ‘Blended Prototyping’.

This work is structured as follows: It starts with the description of the current state of prototyping, both in general and specifically with regard to the prototyping of mobile applications (Chap. 2: Prototyping—State of the Art). After that, the concept, execution, and results of an expert survey are described, which was

conducted to identify requirements for prototyping tools of mobile applications in different development stages (Chap. 3: Requirements Analysis from an Expert Survey). I profited from the knowledge gained in this analysis in the development of the Blended Prototyping platform, which is described in the third chapter of this text (Chap. 4: Blended Prototyping—Design and Implementation). The use of the prototyping approach was tested in a comparative evaluation, where Blended Prototyping as well as two alternative prototyping approaches were used by groups of test users in early development stage design tasks. How the study was conceived on the basis of the previously defined requirements, how it was executed, and what results it produced is described in the fifth chapter of this text (Chap. 5: Comparative Evaluation of the Blended Prototyping approach). In the final and sixth chapter conclusions on the results of my research are drawn, its contributions for research and practice are summarized, and further steps and extensions I imagine for the Blended Prototyping approach are outlined.

# Chapter 2
# State of Prototyping Mobile Application User-Interfaces

This chapter gives an introduction into the scientific discussion and practices of prototyping mobile application user interfaces. For this, it first takes a more general perspective on prototyping (Sect. 2.1), where the term prototyping is defined, its goals are displayed, and different prototyping paradigms are explained. Followed by that, in Sect. 2.2, a particularly relevant approach for this work is displayed in more detail: the approach of a paper-based prototyping. After that, in Sect. 2.3, the text concentrates on prototyping techniques for mobile app user interfaces. Here, especially mixed-fidelity prototypes and their development in different approaches are discussed, and an existing gap in the technological concepts and tools is identified. This motivates the research questions that build the objectives of my work, which are postulated in the Sect. 2.4 that concludes this chapter.

## 2.1 Meaning and Purpose of Prototyping

### *2.1.1 Definition of Prototyping*

The term prototyping is widely used, however, the understanding about its meaning differs widely [162].

*Prototype* is a mixture of the two Greek works *proto* (=first) and *typos* (=impression). Regarding the origin of the word, the term can therefore be understood as any kind of pre-version of a product that serves the purpose of providing a first impression about a system, which does not yet exist.

Putting the term into the context of software development, Sommerville [92] defines a prototype as "an initial version of a software system that is used to demonstrate concepts, try out design options and generally finds out more about the problems and its possible solutions". A similar definition is found by the Institution of Electrical and Electronics Engineers (IEEE), which consider the process of



B. Bähr, *Prototyping of User Interfaces for Mobile Applications*,
T-Labs Series in Telecommunication Services,
DOI 10.1007/978-3-319-53210-3_2

working with prototypes as "a type of development in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analyses in support of the development process" [117].

Developers use prototypes to gain feedback and experience about a not yet fully built system. The prototyping process does not follow a given set of rules: A prototype can be employed to explore whatever information is most relevant to the developer at a given stage. A prototype does not need be polished or finished; it can be of whatever form, which is able to serve the purpose of gaining feedback about design decisions. The earlier such decisions are tested, the easier it will be to regard the generated insights in the further development. This way, prototypes can help to lower the risk of wasting valuable resources on wrong development paths [67].

As described in his Usability Engineering Lifecycle by Jacob Nielsen [107] the prototyping process should be done in cycles of iterative refinements. Each of these refinement cycles produces a new prototype, which is evaluated in user tests or expert evaluations. In user tests the prototype is presented to an exemplar user of the targeted software product. User tests can generate general feedback, specific questions on design implementations, or usage data to identify usability problems in the tested design. Expert evaluations are considered to be a rather inexpensive alternative to user tests that provide another view on the prototype. Here, especially cognitive walkthroughs, heuristic evaluations, or pluralistic walkthroughs are methods that are capable of delivering usability-problems or design flaws on the basis of expert judgments [21, 107, 110].

### 2.1.2 Prototype Information Goals

As prototypes can serve different purposes, different prototyping techniques are targeted at different design questions. For this reason, Szekely [145] provides a good overview of prototyping information goals, that help to get a better understanding on the purpose behind different prototyping approaches. He highlights four major development steps, where prototypes are usually applied: in the *task analysis*, the *user interface design*, in *feasibility studies*, and for the testing in *benchmarking studies*.

Software systems are created for a certain general purpose, which is achieved by supplying its users with a number of different tasks. The general purpose of a system is usually clear throughout the development, the form and characteristics of tasks necessary to fulfill this objective needs to be carefully evaluated. The experience, expectations, and habits of users are diverse, and most of the times deviate from the initial thoughts of system designers. Therefore the concept of an application's task design and it implementation should be tested with users as often and early as possible. Here prototypes play a key role in the ***task analysis*** of a software design, to generate profound knowledge about the system's multiple tasks, and for their implementation in software functionality.

A comprehensive knowledge about necessary software tasks is very important, however, it can only generate real value in the later product, if the software users are actually able to understand and use the implemented methods. This is determined to the largest extent by a suitable ***user interface design*** of the software application. A good interface design is not only related to a nice appearance of the software, but foremost to the way the user finds her way through the different interface screens, software menus, commands, and other aspects that are important in controlling software. A well-made interface design should implement a clear interaction concept, which the user can understand and follow throughout the use of the application. Moreover, the user interface design shapes the users' emotions and attitudes in the software interaction, determining the user experience.

A number of design guidelines are available, that provide a good status quo of common patterns and approaches. However, experience and good advice can never guarantee that a user is able to understand the implemented concepts, is not overloaded with information, and moreover appreciates the design [113]. Here, prototypes play a key role in investigating user interface approaches with user tests. Such tests help to gain valuable feedback about the implemented interface design ideas and provides insights on issues in design approaches early enough, to be still able to adapt the interface concepts accordingly.

Prototypes do not only serve as a communication tool between developers and users, they are moreover essential in investigating technical aspects of the planned software. In ***feasibility studies*** prototypes help to make sure that single technical aspects of the planned software can actually be implemented in the given technical environment. It is rather the rule than the exception that it software development projects smaller and bigger issues in the original planning arise, which might for example derive from an unplanned behavior of employed programming libraries, or complex interdependencies of different program modules. Here, early tests of the software's feasibility, with prototypes that prove certain technical uncertainties, is very important to discover such issues early enough to be still able to change the planning.

Even though the resources and performance of computers is steadily growing, efficiency of the implemented algorithms is always an important aspect. Inefficiencies can add up, and are oftentimes caused by flaws that grow to big problems in the long run. To uncover such inefficiencies prototypes are created and tested in ***benchmarking studies***.

### *2.1.3 Prototyping Paradigms*

The difference of prototyping techniques can be well pointed out in a categorization along a span that is built by two very different prototyping paradigms. Kordon and Luqi [82] point out two opposing approaches: the *throwaway* and the *evolutionary* prototyping paradigms.

*Throwaway* prototyping focuses on employing prototypes with the one and only purpose to generate insights about a design idea. The prototype itself looses its value after the testing, and can be literally speaking thrown away. For not wasting to many resources in the throwaway prototyping process, mechanisms are available to reduce the effort to produce testable pre-versions of an idea to a large extent.

On the other hand, *evolutionary* prototypes are more mature versions of an idea that is implemented with the same tools, which are as well used in the later product development. Evolutionary prototypes are reused after the testing, in a way that they are altered in accordance to the test results and then reused in a new test cycle.

Similar to the development of species, but hopefully quicker, they therefore go through an evolutionary selection process. Finally, an evolutionary prototype ends up becoming the product itself. As a matter of fact, the final software product can even be considered to be an evolutionary prototype for the next software version.

Throwaway prototyping is usually used in rather early design stages, where the system requirements are still vague and not too much effort should be invested to evaluate an idea. In contrast to that evolutionary prototyping is used in rather late development stages, where sufficient insides about principle design decisions exist and the prototype foremost addresses technical and long-term use issues.

In principle, evolutionary as well as throwaway prototypes could be built with the same standard development tools as the later product. Software can be implemented in a state where it is not completed, but already presentable. In fact no software will be programmed without intermediate steps, where the programmer runs the software to check for instant flaws and crashes. Especially regarding the design of user-interfaces, modern programming languages offer tools to achieve comparably fast testable results.

The time where user interfaces were exclusively implemented in writing lines of codes has passed. Most modern programming technologies allow for a clear separation of concern between the design of user interfaces and the implementation of the software functionality. Here, the definition of user-interfaces is handled in separate files that are used to automatically generate the user-interface in the process of compilation or at runtime. Modern standard IDEs (Integrated Development Environments) offer specific interface builders, where the user interface is created and edited in a graphic environment that provides developers with an instant visual feedback (compare Fig. 2.1). Such graphical user interface builders resemble drawing tools in their structure and use, where user controls are positioned on a stage with computer mouse input, and refined in their properties in according dialogs.

Compared to the programming of interface designs solely in code, modern GUI builders are a big advance towards a faster, better designed, and UI-centered development of software. However, it needs to be regarded, that the functionality of the software still needs to be implemented in programming code with a considerable effort. An effort, which is lost when ideas are discarded and the prototype needs reprogramming.

Especially in earlier phases of user-interface design, where principle design questions are yet unclear, the risk of making the wrong decisions is high. Therefore,

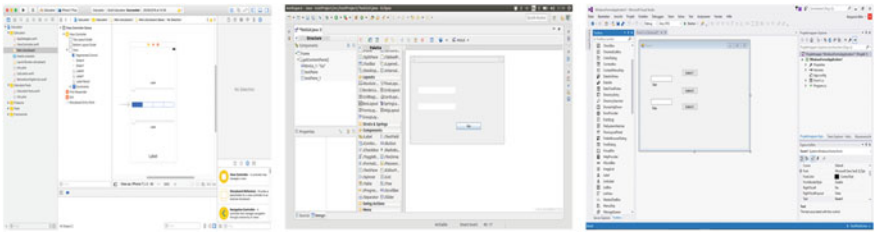

**Fig. 2.1** Modern IDEs that provide visual GUI editors (from *left* to *right* Apple xCode, Eclipse, and Microsoft Visual Studio)

the application of complex high-fidelity IDEs in an evolutionary prototyping approach risks losing valuable project resources in redesigns. Or even worse, but nevertheless oftentimes true, due to time and cost pressure, wrong design decisions are not reverted.

Here, prototyping methods should be applied that allow for a throwaway prototyping fashion, without big investments. Such approaches allow for a rapid development and test of different prototype ideas.

A well-established and radically throwaway oriented technique is found in the *Paper-Based Prototyping* approach. This low-fidelity approach focuses the conception, design, and testing of prototypes on handmade paper sketches. It combines advantages not only in the speed of the prototype creation, but has strengths in advantages for the design team and the prototype testing itself.

## 2.2 The Paper-Based Prototyping Approach

Paper-Based Prototyping (=PBP), as defined by Carolyn Snyder [140] in her work that became a standard introductory reading to the topic, 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 doesn't explain how the interface is intended to work*" (p. 4). According to Snyder, the approach can be used for the prototyping of pretty much every technical system that has user-computer interface, especially websites, web applications, stationary software, or those for handheld devices. The methods of PBP supply techniques for the brainstorming, design, creation, testing, and communication of user interface ideas.

The benefit of sketching on communication, ideation, and creativity is discussed in different research domains. To provide a better understanding on how PBP adapts the advantages of sketching, the section starts with a discussion on the connection between the process of sketching and creative work (Sect. 2.3.1). Followed by that, the function and process of the PBP approach is explained (Sect. 2.3.2), and the key advantages of the approach are displayed (Sect. 2.3.3).

### 2.2.1 The Paper-Based Prototyping Session

The evaluation of PBP prototypes requires the presence of a number of design team members that take different roles in the test sessions. Tests of PBP prototypes should preferably take place in a usability laboratory, where measures are taken to optimally observe and record the user interaction with a prototype. In the testing process, the test *user* is introduced to an idea that is presented to her[1] in the form of paper-sketches. These paper-sketches are laid out in front of the user, with which she is asked to interact as if she would use real software. Using the software is not done with computer hardware input, but is simulated with low-tech techniques. For example, the user could use a pen to point at different points of the user interface she means to click, or keyboard entries are simply described with spoken words. A team member plays the role of the *computer*, which means that she physically changes the user interface just in the way the presented interface would be manipulated on the computer. For this physical manipulation the *computer* can use whatever helps the purpose. Different interface screens are prepared in form of different paper sketches, which the *computer* can swap with another. Dynamic interface content, like checkboxes or dropdown menus, can be simulated with prepared snippets that are quickly positioned on the interface screen. The *computer* is principally allowed to do whatever serves the purpose of displaying the interface functionality. However she needs to follow a well-prepared and well-trained script, so that she does not accidently makes mistakes in the given feedback. This way, the test user is able to interact with the paper prototype idea just as if it was real software, gain her personal experience with the idea, and possibly uncover usability issues. Usually, the user is asked to give think-aloud protocol of her usage, meaning that she talks about the thoughts, expectations, and actions she has in mind.

The person acting as the *computer* is not allowed to talk to the user and explain the measures she takes. She solely acts the role of a real computer, which does not explain its behavior either. The talking with the user does another team member, playing the role titled the *facilitator*.

The facilitator describes the test procedure, presents the user her tasks, and is to be addressed with questions that might occur. The strict separation of the roles of the facilitator and the computer helps the user not to get confused about why the team-member playing the computer cannot be addressed with questions concerning the software feedback.

A disadvantage of the PBP method is in the high effort that is necessary to record and later analyze the prototype test sessions. Unlike in on-device tests, where the user interaction can be tracked automatically to a large extent, sufficient documentation in a PBP session requires a considerable amount of involvement by the development team. Recordings should be made on video and audio to allow for a

[1]Remark on the use of gender-neutral pronouns in this work: I decided to use *she* as a gender-neutral pronoun throughout this work. In my personal view, the *singular they*, oftentimes used as an alternative, creates confusion reading the text.

later analysis of the test sessions. Beside that, *observers* should be employed that continuously write down their impressions about the test sessions. Such human *observers* might be able to capture important events that are not recorded on tape. Moreover, they give the other team-members involved in the test certainty to concentrate on their individual tasks.

Apart from being tested with users, paper-based prototypes can be evaluated in expert reviews, like heuristic evaluations, cognitive walkthroughs, or pluralistic walkthroughs [21, 107, 110] can be applied. Such usability inspection methods are not executed with test-users but with usability-experts, who apply an analytical analysis method to reveal user interface flaws.

### 2.2.2 Advantages of the Paper-Based Prototyping Method

The PBP approach has advantages for the prototyping process in a number of ways. As proposed by Snyder [140] can be grouped in *advantages for the design team*, in *advantages for the development process*, and in *test-user related psychological advantages*.

1. ***Advantages for the Design Team***

The PBP approach uses a number of advantages that derive from the technique of sketching. A good introduction into details of the process of sketching, the psychology of the sketcher, and how the technique can promote creative work is found in the work of Johann Habakuk Israel [71].

Sketching is often referred to as a process of self-communication that is carried out in a procedure of ex- and internalization [71, 123, 130]. The process helps the sketcher to transfer internal ideas in her mind to an external media that is accessible to others. The process leads to a reflection about the externalized content that supports the further creation and materialization of ideas [54, 56].

As Miller [99] points out, the capacity of the human working memory is very limited. Sketching provides a mechanism of *off-lading*, meaning that sketched and therefore externalized ideas are freed from the scarce resources of the working memory. Therefore, the sketchers working memory is freed for other workload connected to the creative ideation process [150].

Consequently, the PBP prototyping approach is addressed to be an uncomplicated and time-efficient alternative to the programming of user interfaces with standard development processes. This helps to produce more prototype results in a faster fashion, and moreover raises the designers' acceptance to discard ideas that proved to be weak [31, 140].

One of the biggest advantages of the PBP approach is the simplicity of its use. Creating sketches with pen and paper is a technique that does need to be extensively trained. The look of PBP prototypes does not have to be perfect, in fact, as described in detail below, rough looking designs oftentimes even perform better in the tests. As a consequence, team members of any professional training can

participate in the design discussion. In fact, those members who do not have a particular expertise in interface design can oftentimes contribute valuable unorthodox ideas.

As an example, staff members who are involved in customer services will usually have valuable experience on the user-interfaces of past products, but do usually not have sufficient skills to participate in software programming focused user interface discussions. Here, paper-based prototyping helps to provide a common base, where team members of different professions can meet at eye level to discuss their product ideas [59, 140].

Even the user herself can be involved into the design process in PBP sessions. They could be invited into the design sessions, or actively participate in the redesign of prototypes during their test sessions. Due to the simplicity of the design process, project manager can have an easier look on the developments without relying on the status support of developers. Moreover, new team-members can be better introduced and participate in the design work [78, 101, 140].

2. ***Advantages for the Development Process***

Especially in early stages, successful user-interface design requires good ideas and creativity. Creativity in the design process hugely benefits from a free design process that does not limit the ideation process with too many restrictions. Considerations about limitations in the applied software technologies, or database structures can very easily disturb the creative flow and should therefore rather be regarded in considerations at later stages in the development process.

Paper is patient; while sketching an idea about an interface idea on paper, no popups will occur, asking for too much detail or informing the designer that control Type X cannot be used in conjunction with Layout type Y. Making a sketch is a native process to sort thoughts and to communicate and develop them with others [71, 123, 130].

Though very skilled designers seem to do magic with their computer software skills, the process of developing a visual prototype on paper is much faster than the process with a computer tool. In fact, to sort their ideas and being able to concentrate on the design tool use, many designers even do quick hand-sketches in parallel to using computer software [31].

When using programming techniques, even very simple prototypes require a considerable amount of effort. Basic features of the software need to be implemented and data the program processes needs to be drafted and managed. Snyder [140] estimates about the effort of coding of about 90%, when creating a prototype with standard programming tools.

The easy way of the PBP approach to design and test prototypes, gives them the freedom to explore different prototype variants, without risking too much effort to be wasted. Therefore, they are not forced to discard promising ideas too early and are more likely to find the best solution. Dow et al. [47] found in a design study with users that exploring multiple prototype approaches at the same time yields in improved design results, compared to a sequential prototyping.

Another advantage of the simplicity of the PBP approach lays in its ability to support flexibility in the prototype testing. If obvious flaws or shortcomings in the interface design are found in the process of a test, the prototype can be quickly adjusted. This can have positive effects, giving the user the immediate impression that her opinion is relevant to the design process. However, such prototype alterations have to be handled with care. Ending up with numerous altered prototypes makes a comparative analysis of their usage impossible.

3. ***Test-User Related Psychological Advantages***

Purpose of prototype evaluations is to gain user feedback about the current state of a user interface design. This feedback is gained from observations and from the users' written or spoken judgment and comments about the tested prototype. Especially on the self-reported feedback by the user, paper-based prototypes oftentimes have a positive effect.

Depending on individual aspects like culture, age, affinity to technology, and character, some users tend to be shy to give an open feedback on prototypes they tested. They might want to be polite and respectful towards the work of others and as a consequence hold back their critic feedback on the approach. Moreover, especially inexperienced users that face problems in the interaction, tend to search the cause of their problems rather in themselves than in the presented software. This reluctance to speak out negative feedback is poisonous to the prototyping process that specifically aims at uncovering weaknesses in the approach that should be solved. The more perfect looking and ready-made a prototype appears, the higher is this reluctance. In contrast to that, rough or even childish looking paper prototypes can lower the respect of the test user towards the design approach. This helps her to articulate negative feedback and suggestions more freely [140, 154].

At this point, it needs to be asked, how test users with high affinity towards computers react to paper software representations. Such users are accustomed to eloquently use complex software products, and might feel treated like a fool being presented with a couple of rough paper sketches. Snyder reports that her research experience does not support these concerns, in the contrary: such expert users oftentimes understood the benefit of the testing method and appreciated it.

Another important advantage of the PBP approach is that it is based on hand-sketches, which have an inherent degree of abstraction. Creating a sketch does not mean reproducing reality in every detail, but is moreover about leaving out aspects that are of less relevance. This aspect makes sketches a great tool to communicate ideas in a more targeted way. Schumann et al. [2] observed, that architects in early design stages prefer to present their ideas to customers in the form of sketches, rather than photo realistic looking rendered computer models. The reason for that is their experience that sketches are a better tool to focus the discussion and feedback of customers to the aspects that are currently most relevant, whereas a perfect looking version of the design state oftentimes distracts with unnecessary detail.

A similar effect is reached with paper sketches in user-interface tests for software. For example, if a prototype is primarily meant to find out whether or not users understand the software menu structure and find all needed methods, it might be a good idea to roughly sketch a paper-prototype in just one color. This way, since users will usually understand that the later software product will not look exactly like the sketched prototype, aspects of the software color design are not part of her feedback.

This way, prototype designers are provided with a great way to focus their prototype on the most relevant aspects of the given design stage, and postpone questions on too much detail.

Diefenbach et al. [44, 45] conducted two different studies, in which they compared how the level of fidelity influenced the result of the prototype testing. The group tested prototypes of different physical objects, like interactive pillows or lamps. Their results were similar to the findings regarding low-fidelity sketches in the software prototyping process: Prototypes reduced in their outward appearance are not less able to produce valuable feedback for the further design process, but are better able than final designs to deliver concept feedback.

## 2.3 Prototyping of Mobile User Interfaces

In the development of mobile apps different issues have to be solved, that play no or a less prominent role in the development of software for the stationary use. This regards aspects like unstable network situations, handling different screen-sizes and —orientations, or a much more strict management of scarce computational resources. However, the development of stationary software and the development of mobile apps are in many ways closely related. It should therefore be considered, how tools for the development of mobile app user interfaces can benefit from the decades of experience made with prototyping methods for the design of stationary software.

For the evolutionary prototyping of mobile apps, the standard development frameworks are very well usable. In fact, since mobile app development technologies are comparably new in general, technologies for a user-interface focused design are well established: modern user interface builders are standard for all platforms, and oftentimes even more advanced concepts like storyboards are provided, where the flow of the user-interface can be defined and viewed.

The adaption of throwaway approaches like paper-based prototyping has its limitations in the design and development of mobile applications.

### 2.3.1 Low-Fidelity Prototyping in the Mobile Context

As explained above, normally PBP test sessions are held in laboratory environment, where the observation of test users can be optimally achieved. Related work gives a lively debate on whether or not such stationary test surroundings can substitute the evaluation of mobile apps in the mobile context.

In fact, mobile user interfaces have to provide solutions to a whole number of specific challenges that derive from their mobile use context, that are presented in detail in the work of Sá et al. in [127] and [125]: Mobile devices have become a predominant part of our daily life. They are not just used in a focused manner sitting at a work desk with optimal lighting and seating conditions, but practically everywhere we go. This myriad of different use contexts produces changing external conditions of distraction, the mobile user interface has to compensate. Lighting conditions are a big factor; in bright sunlight the user interface should be still able to deliver readable screen output, whereas in the darkness of riding a car at night, a user interface might want to use less bright colours for not to distract its user. Other oftentimes-occurring sources of distraction are surrounding noise or the users' changing movements and postures. Such factors hugely affect the user's attention and input precision. Mobile interface need to cope with such factors to allow its users to control the application, whenever they return their concentration to it. For this reason, the interface structure and use logic should be much easier to follow, than this for stationary software. Apart from taking general measures to meet these challenges, some applications dynamically change appearance in reaction to the users' surrounding conditions. Typical examples for the latter are apps for car navigation, which use the device's brightness sensors to switch its user interface presentation between a night- and daylight-mode.

The discussion of the advantages and disadvantages of testing software rather in the laboratory or the field context has been held controversially for several years, especially since mobile devices grew in importance [106].

A number of different studies are found that did comparative studies about tests in laboratory as well as in field test conditions to investigate the information gained from such tests. Here, the terms *testing in field conditions* and *field tests* have to be separated. *Field tests* are commonly used term in software development, which describes tests where potential users of the later software product try out an advanced prototype for a longer period of time [77]. The *testing in field conditions* the articles described in the following are referring to, regard software evaluations that are done in sessions where the test-user experiences the prototype for a limited amount of time, just like this happens in the laboratory context.

It is remarkable, that though these studies are very similar in their proceeding, they delivered very different results that lead to contrary conclusions [48, 73, 77, 106].

For example, Kaikkonen et al. [73] conducted an experiment where the same mobile application was tested in both contexts. Though the authors remark that the evaluation in the laboratory was limited in simulating the mobile use contexts the

app would be usually used in, they describe the necessary effort to conduct user tests in the wild to be very high. In their conclusion they state that the high effort to conduct field test and the challenges in logging the user interaction may not be worthwhile the additional information that is generated about using the interface in different use contexts.

However, in an equivalent study, Duh et al. [48] came to different results. They found a higher number of usability problems in the mobile test context, parts of which they addressed to be of severe importance. Duh et al. identified factors, like noise level, movement, a lack of privacy, or additional stress and nervousness, which often lead to usability problems but cannot be simulated in a laboratory context. As a consequence, Duh et al. underline the necessity to conduct user tests in the field.

Yet another study by Kjeldskov et al. [77] support the previous findings of Kaikkonen et al. and opposes these of Duh et al., Kjeldskov et al. describe the added-value of tests in field conditions to be small. Their findings of usability problems for both conditions were very similar. Moreover, they underline the risk that in tests outside of a laboratory singular events are likely to happen, which cannot be appropriately captured and analysed. In addition to this, they found it harder to capture the concentration and motivation of users participating in studies in the field.

Moreover, they underline the risk that tests made outside of a laboratory are hard to be controlled sufficiently enough, to capture unforeseen events that might affect the user interaction.

The work of Kjeldskov et al. produced a controversial discussion in the mobile HCI community. As a direct answer to the beginning sentence Kjeldskov et al.'s work title "*Is it worth the hassle?*", Nielsen et al. [106] published a paper, which title started with the words "*It is worth the hassle!*". In the paper, Nielsen et al. present findings that contradict the field test sceptic results of Kjeldskov and support those of Duh et al.

Just like Nielsen et al., findings by Sá et al. [125], Brewster [28] and Consolvo [35], share the conclusions of Duh et al. that the ability of tests in the field to uncover a higher total number and more severe usability problems should outweigh its issues in controlling factors like singular occurrences of disturbing events or unsteady user focus in the interaction.

The diversity in the results of the different studies might result from factors like the specific character of the tested app or the events users faced in the different mobile use contexts. However, it has to be highlighted that not all but at least some of the studies report that important usability problems were only found in the test in the field. This should be motivation enough to conduct such tests in the mobile context.

A common sense in the discussion is found in the perception of the high complexity of testing prototypes outside the laboratory. However, it needs to be asked whether better tools should be developed to reduce the *hassle*, which Kjeldskov et al. and Nielsen et al. refer to, rather than to discard mobile tests due to their effort.

Moreover, the referenced authors share the agreement that both testing approaches have their own advantages. Therefore tests in the field should not be seen as a total substitution of tests in the laboratory, but as complementary approaches.

***Low-Fidelity PBP on Mobile Devices***

It is possible to test prototypes developed in a conventional PBP proceeding directly in the mobile context. Paper prototypes employed in device dummies out of wood or plastics were built, where cardboard sheets that represented the device screen, were testes in studies by Hendry et al. [59], Sá et al. [124, 125], or Svanaes et al. [144]. The device dummies used in such studies were carefully constructed to resemble the originals in size, hardware buttons, and weight.

Test users were followed by design team members to observe and react to the usage of the prototype. This way, just like in stationary prototype sessions, the users were able to interact with the prototype by giving touch and gesture commands, which the team member playing the role of a computer regarded with according cardboard-screen changes.

Though this testing process is principally implementable, it has a number of severe limitations. The employed device dummy is not able to reproduce all different effects that occur using real hardware and that can affect the interaction to a large extent. For example, the surrounding lighting conditions will not influence the device interaction in the same way. Content on paper can be perfectly read in bright sunlight, not so a mobile device screen [124, 125].

Moreover, the device input can only be poorly simulated. Whether or not control was actually targeted by a user's touch input can only be estimated. Using a simulated on-screen keyboard out of paper will likely produce no typos. Especially in mobile use contexts that involve movement of the user, however, such usability aspects are highly important.

### *2.3.2 Mixed-Fidelity Prototyping Approaches*

Both, low-fidelity as well as high-fidelity prototyping approaches have their advantages and limitations for a prototype driven design and development process. Comparing the two approaches shows that they complement each other, meaning that most aspects that are regarded as weaknesses for the one approach, are considered as a strength of the other one.

As a bottom line it can be stated that in very late development stages, where the evolving code base of the product development is of central interest, the application of an evolutionary prototyping approach with high-fidelity prototypes is advisable [82, 97]. In contrast to that, in very early stages, where a free collaborative design towards quickly testable results is key to success, a throwaway approach with low-fidelity approaches like the Paper-Based prototyping, should be the approach

of choice. This conclusions leads to the question on how prototyping should be performed in the middle phases of the development.

As an answer addressing this question, a number of techniques have been developed that facilitate a *mixed-fidelity* prototyping approach for the design and development of mobile app user interfaces. Such techniques aim to adapt the advantages of both, the Low- and High-Fidelity approach, and at the same time to overcome their limitations. Coined by authors like Sá et al. [127], McCurdy et al. [97], and Coyette et al. [37] the term mixed-fidelity prototypes refers to techniques, which produce prototypes that run as an application directly on the target device, but still keep the rough looking appearance of paper prototypes, in displaying the created sketches directly on the screen. These hand drawn, or hand-drawn looking, interface prototypes are provided with functionality, so that they are able to react to the user's input commands in changing the shown screen on the device.

Displayed on the actual device's screen, mixed-fidelity prototypes can be used to examine a bigger number of usability problems than those of a low-fidelity. Factors like screen related disturbances (i.e. lack of brightness or mirroring) and the user's input precision controlling the touch screen occur in the mixed-fidelity prototype just as in the later app.

Unlike paper prototypes, where a team-member has to act as a computer in manipulating the prototype in the testing, users can test mixed-fidelity prototypes autonomously on their devices. In a low-fidelity prototyping session, the member taking the computer-role has to constantly search for the right screens to put in the drawer, which gives the interface a tremendous reaction delay. An observer-role-player always has to have one eye on the devices screen and the other one on her notepad, writing down her observations, which typically makes her miss interaction aspects or forget to protocol them. Logging algorithms running on a device will record the data they are appointed to with higher reliability.

Witnessing the test scenario directly, human observers may be able to recognize events that could not be reproduced from the post-analysis of video data. For example, this regards the user's emotional state, or suggestive forms of communication she might have with other persons in the test room. However, especially if applied in a stationary test context, human observers could be employed in tests of mixed-fidelity prototypes as well.

In a comparative study, Lumsden and Maclean [95] investigated the testing of low- and mixed-fidelity prototypes in the mobile context. They found, that mixed-fidelity prototypes were able to identify a generally higher number of usability problems, as well as to uncover more issues that were rated to be severely important to the further user interface development, primarily regarding its terminology and usage-paths.

In the following, mixed-fidelity prototyping approaches in three different application domains are displayed: mixed-fidelity approaches on *desktop computers*, on *mobile devices*, and on *interactive surfaces*. Each of the descriptions are segmented in two sections: first examples from the field of research are provided, then examples of mixed-fidelity prototyping applications in commercially available tools are displayed.

#### 2.3.2.1 Mixed-Fidelity Prototyping on Desktop Computers

***Examples from Research***

Already in 1995 an approach was developed by Landay and Myers, to convey PBP to desktop computers, in providing a sketch-based user interface design tool named SILK (*Sketch Interfaces Like Krazy*) [70, 87–89]. Not surprisingly, since already developed in the 1990s, the software supported only the development of user interfaces for a stationary use. However, the concept is pioneering work and could be easily adapted for the design of mobile applications.

SILK uses a graphical tablet connected to the computer, to allow its users to digitally sketch interface ideas. Pen-stroke recognition algorithms are implemented in the software that interprets the sketches for pre-trained strokes to define user controls. This way, buttons, check boxes, and alike can be quickly defined by the designer with a simple pen gesture. The other drawn content is kept in the interface in form of the original sketch.

The user can open two different types of views on the developing prototype: a sketch view, where a life-view of sketched content is displayed, and a storyboard view, where connections between the sketches can be defined. To define such connections, the graphical tablet is used in a stroke-gesture reaching from a previously recognized button to a target screen. Connections are displayed in the storyboard view as arrows, providing the designers with an overview of the interaction flow of the created prototype.

SILK allows the testing of the developed prototype in a prototype player at the computer. Here, the degree of fidelity the prototype displays can be selected manually. SILK offers views on the prototype, where the sketch and its intrepreted version are displayed next to each other. Elemets of the sketch that were not interpreted are shown in their raw form in the prototype. This way, a mixed-fidelity prototype design is created.

The makers of SILK already thought about the reusability of the created designs: Designers were able to export their recognized interface versions to Visual Basic 5 or Common Lisp programming code.

SILK adapted the PBP approach in several ways. Though not made on physical paper, the user interfaces were created on the basis of sketches. Moreover, giving the opportunity to test in a low-fidelity view, SILK facilitated the advantages of rough-sketched looking prototype designs for the testing process.

The storyboard view of SILK did not only allow to define linkage paths, but also provided designers with an overview of the prototype flow. However, in prototypes consisting of many screens, the storyboard-view quickly had issues displaying the whole screen range.

An evaluation of SILK with 12 test-users showed that the technique effectively supports the design of UIs in early stages, and that the tool also provides an effective way of communicating design ideas between designers and engineers [89].

In a further development on the basis of SILK, by Newman et al. [105], developed a new tool named DENIM (Design Environment for navigation and

information models). The expression *informal* was used in tool title, since it was attributed to develop interfaces that are "designed to support natural, ambiguous forms of human-computer interaction" [89].

Just like SILK, the DENIM tool was based on the input with stylus input devices, connected to a stationary computer. Using a newly developed stylus-interaction library called SATIN [63], the electronic pen in DENIM could be used with enhanced gesture recognition techniques.

This allowed DENIM to establish a fully zoom-able design stage, where users could look at their prototype in different forms of perspectives. Each of these perspectives has its own purpose in organizing and creating the developing interface. In the Detail and Page view, the single pages are in focus with the sketching of their content and definition of their functionality. The Storyboard view serves to build connection links between certain user controls and connected pages, whereas the Site Map view and Overview is used to outline and create the interfaces screen segmentation.

An evaluation of DENIM with seven professional designers found that the tool was easy to learn and understand, however not particularly easy to use. However, Newman et al. point out, that the system's ease of use will likely improve with better performing Tablet PCs, than those available in 2003 [105].

#### *Examples in Commercial Applications*

Different commercially available approaches for the development of mixed-fidelity approaches exist. They provide designers and developers of mobile apps and other computer software, especially websites, with well-developed tools that are very frequently used in the professional context of user interface design.

#### Balsamiq

The software Balsamiq [166] concentrates its design process on the ideation phase, providing a desktop-computer software, where an interface mockup can be quickly created in the fashion of a usual interface builder. Balsamiq provides 77 different user controls the designer can position on single interface screens. These 77 different controls include a number of very advanced and interactive widgets that are well used in modern apps, but are normally too complex to be implemented in early prototypes.

These include for example maps, dynamic menus or even iTunes like cover browsers. The library of controls can be extended with templates and own controls created by the user. Designers can switch between two different views on the prototype: a *sketch skin* and a *wireframe skin*.

The *sketch skin* provides a sketched look-and-feel, mixed with high-fidelity components like images or videos (compare Fig. 2.2). The *wireframe skin* shows a higher-fidelity view at the design, where lines are straightened and the typical graphical representations of controls is provided, which designers are used to from using wireframing stencils.

Balsamiq prototypes are tested directly on the target device. For this Balsamiq provides a testing infrastructure, where prototypes are made available to the users as

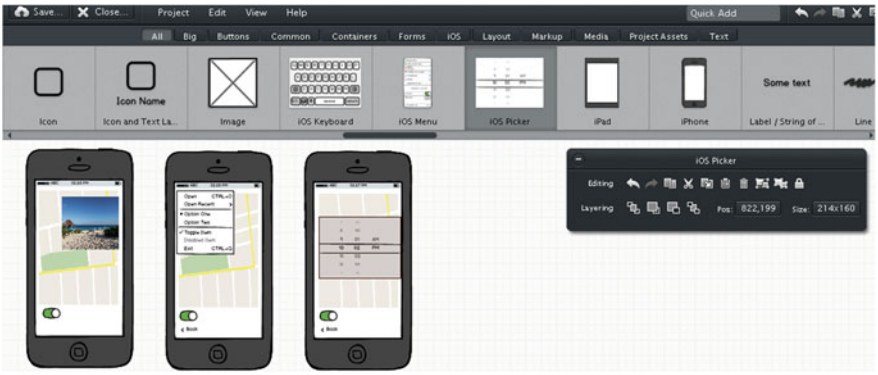


**Fig. 2.2** Balsamiq editor in sketch-skin view

websites that open up in full-screen. In the testing, the designer can determine whether a sketch-like or a wireframe-like version of the prototype should be provided to the test user. The test-platform includes the insertion of questionnaires that are presented to the test users before or after the tests.

The functional possibilities of Balsamic prototypes are limited to the extent to simple click-through versions of the user interface, where the press of a control addresses single path screen changes.

### Axure, MockFlow, and Allikes

Other software tools, like Axure [167] or MockFlow [168], provide professional solutions for the development of user interface prototypes that allow for a higher degree of functionality. The concept of the approaches is very similar, wherefore the following descriptions concentrate on the portrait of one of the tools: the software Axure.

Similar to Balsamiq, prototypes in Axure are designed in interface builder view, where different screens are created and user-controls on which user controls, images, and labels are positioned by mouse, to express the user interface design. Compared to Balsamiq, the number of available user controls is more limited in Axure. However, the possibilities of Axure to determine more advanced interface behavior are far more elaborated than those of Balsamiq.

In Axure the actions of user controls can be edited within dialogs that allow the definition of condition-based rules (compare Fig. 2.3). Here, simple if-then-else cases can be defined in a graphical editor. These conditions are translated into a simple Axure-specific programming language, which can be viewed and as well edited by the designer. This way, skilled designers can enter functional aspects directly in the form of programming code. In such code, not only the widgets properties can be accessed. The definition of prototype variables is possible as well as computing data entered into the prototype, like mathematical calculations or string operations.

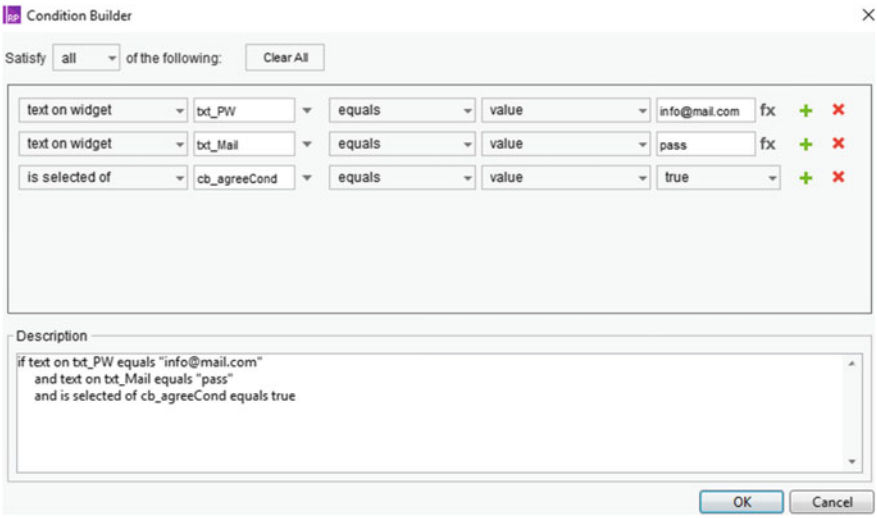


**Fig. 2.3** Condition builder in Axure

#### 2.3.2.2 Mixed-Fidelity Prototyping on Mobile Devices

***Examples from Research***

Sá and Carriço [126, 127] present a prototyping tool that shifts the prototyping of mobile app user interfaces to mobile devices, while providing the additional opportunity to edit the prototype data on a stationary computer. The prototype consists of sketches that can be created directly on the device with a stylus pen, or be created in form of hand sketches on paper. Physical paper sketches can be imported either with a scanner connected to the desktop computer software, or by using the mobile device's camera.

Hand drawn elements can be replaced with interactive components, letting the prototype fluently evolve to a higher fidelity. For these interactive components, linkage paths to other screens or pop-up windows can be defined.

In a prototype player, the tool of Sá and Carriço allow the test of the prototype, directly on the mobile device. The prototype player offers to switch from the testing view to a mode, where the prototype data can be edited. This way, interface flaws can be corrected in the situation of the test. The adjustment possibilities include both, the editing of the embedded user controls, e.g. by resizing, moving or deleting it, as well as the editing of the screen image themselves. The tool includes a logging tool that allows reconstructing after the testing, when and for how long the test users stayed on which screen. The described evaluation results underline positive reactions of the involved designers on the ease-of-use of the framework, as well as its amount of available features. Moreover, test results validated a positive influence of they prototyping and evaluation framework on the design process [127].

### *Examples in Commercial Applications*

Different mobile apps for the mixed-fidelity prototyping of user interfaces on the mobile device exist. Most prominent examples are the POP App [169] and the Marvel App [170]. Both apps are very similar in their functionality and distinguish themselves more in the parallel provided web-interfaces, where prototype data can be edited on a desktop computer. The following description focuses on the more advanced capabilities of the Marvel App.

Just like in the idea of Sá and Carriço explained above, the Marvel app allows its users to design user interface prototypes on the basis of regular paper sketches, which can be imported in photos with the mobile device's camera. Now being available as a digital photo on the device screen, the designer can position active areas on top of the sketches and link them to other photographed screens (compare left image in Fig. 2.4). This way click dummies can be created rapidly on the basis of hand-sketches. Uploaded to the Marvel-App database, the prototype can now be explored by test-users on their individual devices. Similar to the Pop-App described above, the functionality of the prototype in the Marvel-App is limited to one-path screen changes.

Growing rapidly in its supplied functionality since its release in 2014 ago, the Marvel-App now does not only support the prototyping of mixed-fidelity prototypes on mobile devices, but progressed its web-application in multiple ways to enhance the possibilities of prototyping at the computer (compare middle image in Fig. 2.4). Here, similarly advanced user widgets like in Balsamiq are provided to design prototypes of an advanced-fidelity. Their look and feel is advanced with visual transitions to an extent, where the prototypes appearance resembles completely programmed mobile apps (compare right image in Fig. 2.4). Furthermore, the Marvel web-application supports remote collaborative design, allowing users at different computers to simultaneously discuss and edit a prototype. However, programming of more advanced functional aspects is still not supported.

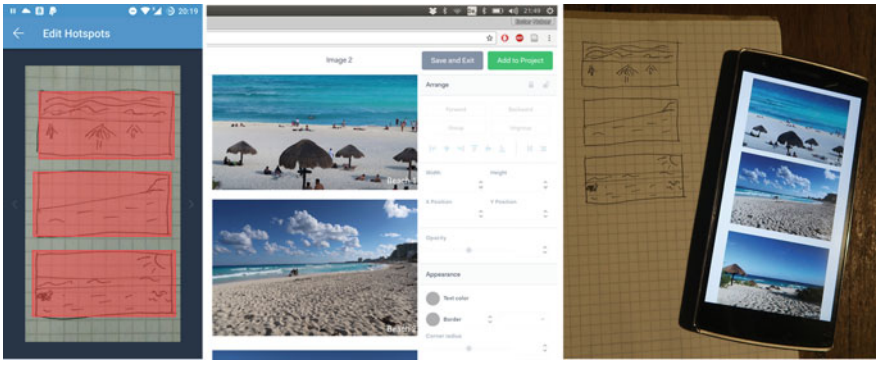


**Fig. 2.4** Marvel mobile App (*left*), web-application (*middle*), running high-fi prototype (*right*)

#### 2.3.2.3 Mixed-Fidelity Prototyping on Interactive Surfaces

In his pioneering work, Mark Weiser [159] explored possibilities of using tabletop computing devices already in the early 1990s. His research generated a great deal of attention and motivated numerous others to investigate the field. Consequently, shortly afterwards first concepts on interactive tabletop systems, like the DigitalDesk [160] or the ActiveDesk [122], were contributed to the scientific debate.

This opened a new field of research on tabletop systems and interactive surfaces, which is still vividly explored. International conference like the one on *Interactive Tabletops and Surfaces*, which will have its 11th venue in 2016, underline the increasing relevance of the field in the scientific community.

With progressing technological capabilities to more and more affordable prices, a big number of different approaches towards this extended interface concept were created. Klemmer et al. [80] built a prototyping platform that conveys the design environment to an interactive wall. Their project called 'The Designers' Outpost' is conceived around a hardware setup that uses a silver screen as the interaction focus, on which designers can stick PostIt notes to collaboratively compose a website's sitemap. Through rear projection and camera techniques the system helps designers to organize and digitalize their concept and lets them define interdependencies between the sitemap elements. It additionally supports the designers with elaborated design history functionality and integrates concepts for a remote collaborative work.

The digital sitemap overviews generated in this fashion generated digital sitemap overviews can be imported into the described DENIM system and serve as a foundation for the further prototypic website development. Therefore the Designers' Outpost does not create testable interfaces, but gives designers a helpful tool in designing first content related concepts. However the interface design process itself is not supported.

Comparing different related work on the topic makes clear, that most approaches aim to facilitate similar advantages of the technology to yield comparable goals. Oftentimes this regards the enhancement of the interaction by *involving physical objects*. Moreover the technologies are widely used to *promote collaborative work* and to *support the creative process*. These three objectives are discussed in the following.

##### *Advantages from Involving Physical Objects into the Interaction*

Already in earliest applications of interactive tabletops, physical objects were included as a part of the interaction scope. With his DigitalDesk [160], Wellner developed an overhead projected tabletop system, which included the use of regular paper sheets in typical office tasks. With the help of the overhead projection, the setup was able to project digital content onto these paper sheets or the table surface. Moreover, the approach already implemented ways to record content drawn on the paper sheets, wherefore the paper was able to serve as an input/output channel to the computational system.

Other authors explored the use of three-dimensional objects in the interaction. Underkoffler and Ishii [151] for example created a tangible tool for urban planning, embedding physical models of building into the interaction. Piper et al. [116] explored possibilities of three-dimensional physical input, in using modeling clay in the design of landscapes in a tabletop-computing environment. Moreover Zufferey et al. [165] created an interactive surface with physical models of storage shelves, which was used for the collaborative planning of storage logistics.

Spindler et al. moved the interaction space above the surface of a tabletop computing system. Their PaperLens [142] system allows users to position and move cardboard pages above the surface, where upon the according section of a three dimensional model is projected on it. For example, this allows users to dynamically browse through a human body in exploring 3D CRT data.

The inclusion of mobile devices into the interaction space of digital table systems has been widely discussed as well. For example, Wilson and Sarin [161] or Shen et al. [135] use a tabletop system to help to explore, edit, and share digital content stored on mobile devices collaboratively.

***Advantages in the Promotion of Collaborative Work***

Computer tools revolutionized the way of today's communication. However, when people are working collaboratively on a project, face-to-face meetings remain extremely important [114, 120].

Such meetings between team members usually take place at tables, which support collaborative work between humans in a whole number of ways. Tables had always been used as a natural surrounding for groups to come together, sit down, and share a common space. People sitting around a table usually take seating positions where they can look each other in the eye and are able to use the tabletop of a large shared workspace. This workspace gives people a natural environment to manage information in the space of the tabletop. Here for example paper sheets can be put rather in the middle of the table to be shared with others, however, at the same time people have a personal space to work on their own thoughts. In contrast to that, computer devices usually focus on individual control and are less well applicable group work [18, 131, 134, 164].

Pursuing the promotion of face-to-face discussions, Tandler et al. [148] created a setup with multiple adjacent touch-screens, which are each controlled by one user. The screens are connected with another, providing the users with the opportunity to access digital content in their personal space, but as well to share information with others by moving it from their own screen to this of a team mate. The DynaWall by Streitz et al. [143] is another surface system that promotes the exchange in collaboration with a big interactive wall. The InteracTable by the same research group, allows teams to use a shared tabletop environment to collaboratively access and edit digital content. Similar applications for tabletop-computing environments for group work were developed, to collaboratively search image databases in TeamSearch [100], or to conduct shared searches in the web with WebSurface [149].

Putting their attention on the advantages of tabletops to facilitate communication, Shen et al. [136] employed an interactive tabletop for storytelling between

users. Apted et al. [6] developed another interactive table that aims to ease the showing and talking about pictures in a family context, which specially regarded needs of elderly people. Using tabletop computers to grant users with special needs access to computational devices, is as well the main motivation of Battocchi et al. [17], who created a puzzle game specifically designed to be used with children with autism spectrum disorders.

***Advantages in the Support of Creativity***

Being a complex high-level cognitive process, creativity is investigated by researches in a whole range of scientific disciplines, including psychology, engineering, or human-computer-interaction [30].

James Blinn [24] describes that the creative process happens as a succession of two phases. In the first phase, fuzzy thoughts are drawn from a state of chaos into gro form of ordered ideas. In the second phase, these ideas become materialized in a presentable documented form. Haller et al. [57] remarks that most computer design-tools target the second of these phases, and sees a need for tools that are better able to allow chaos and the development of ideas. They see computational surfaces and tabletops as an ideal platform to develop ideas from a fuzzy version to a better-structured and concrete form. Ben Shneiderman postulates four design principles for creativity supporting tools: to *support exploratory search*, *enable collaboration*, *provide rich history-keeping*, and allow to *design with low thresholds, high ceilings, and wide walls* [137]. In respect to all of these attributes, interactive surfaces are privileged in comparison to ordinary computational devices.

Buisine et al. point out [30] that although creativity should be considered as an individual capability to a large extent, it can promoted, as well as hindered, by the surrounding working conditions, especially in a group work context. Regenbrecht et al. [119] underline, that for the work on creative ideas in a team, it is key to enhance communication and collaboration. As displayed above, these factors are largely addressed with tabletop-computing systems.

Klemmer [79] and Streitz [143] share the view that ordinary computer tools often fail to promote creative team work, since they distract the thoughts of its users too often with alert boxes of lengthy menu dialogs that require their attention.

According to Kelly [75], additional requirements to promote creative work in teams are to allow members to express their ideas to another both: verbally and non-verbally. The latter should involve natural techniques of expression with physical real life objects, such as pen and paper in hand sketches.

Such conditions can be much easier achieved in interactive tabletop systems, where physical objects can seamlessly embedded into the interaction process, than in ordinary computer systems. Hilliges et al. [60] show this in a collaborative user study, were interactive tabletop environments proved to supply improved creative working conditions.

Different authors developed tabletop-environments that facilitate established creativity enhancement methods in a teamwork context. A mind-mapping tool was implemented by Buisine et al. [30], an approach to promote brainstorming processes and decision making was developed by Hunter and Maes [68].

The examples above show that tabletop-systems are frequently applied to promote both, collaboration and creativeness in different work tasks. A successful prototyping mobile app user interfaces largely depends on these factors, wherefore the use of interactive tabletop systems is a promising approach to improve the process of mixed-fidelity prototyping. Moreover, an embedding of physical objects into the interaction process of tabletop computing devices is easily achieved and perceived as a natural form of interaction.

This allows for the seamless use of physical pen and paper sketches, which have their own advantages for the collaborative design process, as explained in the following section.

### 2.3.3 *Influence of the Sketching Media on the Prototyping Process*

Most of the computational prototyping systems discussed above allow the development of prototypes on the basis of sketches. Some of the presented approaches use digital tools for the sketching process, yet others let designers create their sketches using physical pen and paper.

The digital sketching approaches have two main advantages: First, unlike ordinary paper-pen sketches, digital sketches can be instantly processed digitally, without further photographing or scanning. Second, digital sketching devices work well in the recognition of sketched objects or hand-written text. Algorithms employed in this domain work much better in the analysis of the pen-stroke history, than in the analysis of a completed picture.

However, the traditional paper pen sketching process has its own advantages that motivate a careful consideration about the approach of choice in the development of new technical systems.

First of all, using physical pen and paper for sketches is a native way of expression. Tablet sketching approaches do their best to adapt this process, however, they require a certain experience of the user with the system. Cook and Bailey [36] did an investigation of the physical sketching process and on how well device based sketching techniques are able to provide a substitution. They interviewed a number of designers about the topic and observed their working habits in the sketching process. They found, that physical sketching is still an essential part of the designers' daily routines and that designers feel less free working with digital sketching mechanisms. Similar results were yielded by Newman and Landay [104], who particularly underlined their observation, that the use of physical paper sketches plays a predominant role in design exploration phases, where designers search for free mind to develop their ideas and do not want to lose time on unnecessary details.

Reasons for this preferred use of analog sketching in early design are provided by Cook and Bailey [36]. They argue that a number of tasks essential to this state of

design can be better promoted by the use of pen and paper. Paper is found to be better able to communicate early design ideas in a collaborative ideation process.

When designers share one device for the sketching process a gatekeeper problem is created, meaning that solely one designer has the control to edit content. If multiple devices are used in the process, the sharing of information between groups of designers is done less fluently and the communication-frequency is reduced. Moreover, designers are usually faster in doing hand sketches on paper, wherefore they feel more free to use the physical technique in the context of brainstorming processes, where the quick development and suggestion of ideas in team work is essential. In the conclusion of their study, Cook and Bailey give a clear recommendation to facilitate the identified advantages of physical sketching, rather than to try to substitute them with digital sketching tools.

The advantage in speed of physical tool to create sketches was observed by Nagai and Noguchi [102] as well. Moreover, they found that paper is better facilitating a rapid design development, where different design ideas compared creating quick throwaway design alternatives. Similar observations were made by Vinod Goel [53], who states that designers generating their ideas quickly with freehand sketches, rather tend to explore several variations, than to focus on the refinement of their first designs.

Beyer and Holzblatt [20] underline the advantage of paper to be easily reviewed collaboratively in the real world environment. Grouping paper sheets to sort information, or to spread them out on tables or walls to get a better overview on the matter, are often-used techniques bound to the medium. Of course, computer designs can be printed out and therefore transferred into the same materiality, however, this can mean an interruption on the creative flow.

### *2.3.4 Comparison of Mobile Prototyping Approaches*

In order to compare the prototyping approaches displayed above, and established set of requirements would be useful. However, the above discussion of different prototyping approaches underlined that the needs addressed by different prototyping tools are quite heterogeneous. Authors usually limit their discussions on specific factors that are relevant to their individual work. A systematic approach to develop a comprehensive set of requirement categories does not exist in related scientific work, or as an industry standard. A comprehensive taxonomy does not exist, nor can clear design-guidelines for the development of mobile app UI prototyping tools be found.

The comparison of low- and high-fidelity prototyping tools above points out that the requirements the technologies address change at different stages of the product development cycle: Where low-fidelity approaches are foremost attributed with advantages is the support of a fast prototype creation in group work, the strengths of high-fidelity approaches are primarily seen in the support of reusable programming

code, in tests in the real use-context, or in their ability to regard usability factors in the tests that derive from the mobile devices' hardware limitations. As a consequence, the yet to be developed requirements catalog should take the changing relevance of requirements at different project phases into account.

To allow for a targeted development of new processes and tools for the prototyping of mobile user interface prototypes, the development of such a requirements catalog should be one of the objectives of this work.

In the above discussion of different prototyping approaches, a number of single requirement-factors are repeatedly regarded. The factors are not evaluated, therefore, they cannot be judged in their importance towards each other. However, a comparison of the tools along these factors points out a gap in existing solutions, which should be regarded in the development of the new prototyping techniques addressed in this work.

The scores given in the table above are estimates, made on basis of the discussion of the approaches in related work, as well as on personal experience with the tools. They should therefore not be considered as a formally evaluated result, but serve as an orientation for the following discussion.

As displayed in detail above, the paper-based prototyping approach primarily has its advantages in providing a quick and easy to learn technique that can be well applied in interdisciplinary group work sessions. Moreover, it facilitates the described advantages of a rough looking design that allows for a natural abstraction of the presented content. The shortcomings of the approach are primarily in its disability to provide reusable programming code, its disability to implement more complex functionality in the prototype, its impracticality for large user tests, its limitations to be applied in the mobile use context, and its disability to give information about hardware limitations in the testing (compare Sect. 2.3.1).

Diametrically opposed are the strength and weaknesses of standard development IDE's, employed for an evolutionary high-fidelity prototyping. Here, reusable code, the possibility to implement prototypes of advanced complexity, and the regard of hardware limitations are inherent advantages of the method. User tests in the wild are possible even at large scales, however, measures have to be taken to distribute the prototype and log the user interaction with the prototype. Here, beta-test technologies like TestFlight,[2] TestFairy,[3] or the HockeyApp[4] can be applied. The major weaknesses of standard development tools lay in those areas, where paper-based prototyping approaches play out their strength.

The mixed-fidelity approaches, investigated above take a middle position between these two approaches. First of all, in comparison to standard development methods, they offer improvements in speed and learnability. In comparison to the paper-based procedure, they have a privilege in being able to produce prototypes

[2] www.testflightapp.com (last accessed 11th April 2016).

[3] www.testfairy.com (last accessed 11th April 2016).

[4] www.hockeyapp.net (last accessed 11th April 2016).

that can be ran on mobile devices and therefore allow the tests of prototypes in the real use context, easier tests with large groups, and to regard hardware limitations in the tests as well. However, the described mixed-fidelity prototyping approaches have their own shortcomings.

SILK and DENIM are approaches that had been released long before the first iPhone entered market and the impact of mobile devices began to rocket. They are targeted at the design of websites. However, they could be rather easily adapted for the development of mobile apps as well. For this reason, the ratings for the approaches related to tests on mobile platforms in Table 2.1 are put in brackets. SILK and DENIM allow sketching in their design processes, however, are limited to sketches done on tablet devices. For the reasons discussed above in Sect. 2.3.3 the sketching on tablets is not able to generate the same advantages as sketching with paper and pen, like leveraging group work, creativity, and higher speed in expressing drafts. The technologies allow the generation of Visual Basic user interface code, which can serve as a basis for functionality enhancements. However, the techniques themselves are just able to produce single-path click dummies, where each click on an interactive widget leads to just exactly one target.

Moreover, SILK and DENIM are software tools for desktop computers, which inherently limit the progress of collaborative work. As pointed out above in the discussion of advantages of interactive surfaces for the promotion of collaborative work, different authors underline that regular computer setups weaken the participation of team members, since the design stage of such systems cannot be accessed jointly.

This issue is shared by tools like Axure/Balsamiq/Mockflow and alike, which are also operated on regular desktop computers. However, such professional software implements processes to advance the prototype functionality with short code snippets, without raising the complexity of the tools to an extent, where it is hard to produce simple prototypes as a beginner user. Unfortunately, the programming code used in these technologies is usually based on web-script languages, or on a software specific pseudo-code. For this reason the reusability of the code in the later product development is very limited.

Compared to SILK, which at least allows including tablet-device sketches in the design process, mockup software solutions like Axure discard hand sketches entirely from the design process. Here, user interfaces are created in a manner very similar to standard user-interface-builders. However, the interfaces are then rendered with a sketch-like looking appearance. For this reason, the approaches adapt the advantages of paper-based prototyping for the testing, however, its advantages for the design-process and the design-team that stem from the simple use of physical paper cannot be exploited.

In contrast to that, mobile apps like the one from Sá and Carriço, POP, or the MarvelApp, put the physical use of paper into the center of the design process. Here, paper sketches are physically created and then transferred to the digital space, simply by photographing them with the mobile device camera. Therefore, such approaches facilitate the advantages of using physical pen and paper in the design

**Table 2.1** Estimation on different prototyping approaches to meet prototyping requirements

| | Paper-based prototyping | High-Fi/IDEs | SILK/DENIM | Axure/Balsamiq/Mockflow | Sá/POP/Marvel App |
|---|---|---|---|---|---|
| Quick PTs | + + + | − − − | + + + | + + + | + + + |
| Groupwork | + + + | − − − | − | − − | + + + |
| Easy to learn | + + + | − − − | + + + | + + | + + + |
| Interdisciplinary | + + + | − − | + + + | + + + | + + + |
| Abstraction/rough design | + + + | − − | + + + | + + | + + + |
| Physical pen and paper | + + + | − − − | − − | − − − | + + + |
| Reusable code | − − − | + + + | + + | + | − |
| Complex functionality | − | + + + | + | + | − |
| Large scale tests possible | − − − | + + | (+ + +) | + + + | + + + |
| Tests in real use context | − − − | + + | (+ + +) | + + + | + + + |
| Regards hardware limitations | − − − | + + + | (+ +) | + + + | + + |

process, and will likely take profit from the advantages addressed to the paper-based prototyping approach for the design-team and the design-process. However, a big disadvantage in the approaches is that the degree of functionality that is implementable in the prototypes is limited to one-path click dummies. Therefore, designers using these techniques can quickly come to a point, where the questions they need to address cannot be transported by the prototype.

A prototyping approach is missing that takes use of the advantages of designing mobile user interface prototypes on the basis of physical paper sketches, and at the same time allows for enough functional complexity in the created prototypes, to stay relevant for more than just the first few iteration cycles. Therefore, the delivered prototypes should leverage the advantages of a rough looking abstract design, but as well allow its user to program functionality to whatever necessary extent. For not wasting the efforts done in the programming, the tool should moreover provide mechanisms that make the code reusable in the later product development. This way, the approach should be able to *blend* the paradigms of a throwaway and evolutionary prototyping: Facilitating the advantages of throwaway like design fashion, but at the same time supplying mechanisms for an evolutionary development of programming code.

Such an approach could be implemented in a similar fashion as the discussed prototyping apps, using mobile devices to digitalize the apps and specific basic functionality. However, as discussed above in Sect. 2.3.2.3, interactive surface can provide big advantages for design tasks in groups that require additional digital manipulations. As in the work of Klemmer et al. [80], described before, interactive surfaces can serve as an excellent platform to collaboratively design, explore, and progress creative tasks. Providing a new prototyping approach in the context of a tabletop computer environment could therefore implement an extended information and interaction space that combines the advantages of the physical and digital world a productive design tool.

## 2.4 Research Objectives

The research I conducted for my doctorate aims to find processes and tools to improve the prototyping of mobile applications. As explained above in Sect. 2.3.4, current research lacks a catalog that displays the understanding of the requirements that today's designers and developers address at prototyping systems for mobile applications. A comprehensive knowledge about such requirements, and how they change in different development phases, is however necessary for two reasons: First, to provide a guideline for the implementation of new prototyping techniques. Second, to create a basis for the systematic evaluation of prototyping tools by supplying a set of the most relevant reference points an approach should meet.

Therefore, one goal of my research is to identify and evaluate a catalog that displays the most important requirements designers and developers of mobile apps address for prototyping tools in their domain.

As underlined above, in the consideration of prototyping paradigms, their information goals, and applied techniques, the process of prototyping usually changes throughout different development stages. Where in early stages throwaway prototypes are applied to deliver fast testable prototypes that gain a more general feedback, in late stages evolutionary prototypes are employed, to clarify in-depth questions on specific user-interface aspects. As a consequence the catalog searched for in this work should not only point out the requirements, but also point out their specific relevance at different development stages. This motivates my first research question:

***Research Question 1*** *(RQ 1):*

How can a requirements catalog on prototyping systems for the development of mobile app UIs be found, that reflects the changing importance of requirements at different project stages, and serves the following two purposes:

1.1. to provide an orientation to develop new prototyping tools that meet the actual demands of mobile application developers and designers
1.2. that can serve as a basis for the evaluation of prototyping tools

The development and evaluation of such a catalog is described in the following chapter. Furthermore, the derived catalog should be employed, to test its applicability for the evaluation of prototyping tools. This translates into my second research question:

***Research Question 2*** *(RQ 2):*

2.1. How can this requirements catalog be applied, to evaluate prototyping tools with domain experts?
2.2. How can this requirements catalog be applied, to compare the performance of different prototyping approaches in an application study?

The Question 2.1 is answered in an expert rating of a prototyping approach with the developed criteria, described in the Sect. 3.2.3. The Question 2.2, on how such an criteria catalogue can be applied in a comparative performance study of different prototyping approaches is discussed in Chap. 5 of this work, where a comparative evaluation of Blended Prototyping with two alternative techniques is discussed.

In the center of this work stands the conceptualization, development, and evaluation of a new prototyping approach, called Blended Prototyping. Blended Prototyping should consider the most relevant categories identified in the newly developed requirement catalog as a design guideline, to fill the gap of existing mechanisms in early to middle development stages.

Paper-based prototyping is an established approach that proved to be of high value for the development of early design stage user interface prototypes. Therefore, the Blended Prototyping approach provides processes and tools that aim to improve the prototyping process, by taking full advantage of the strengths that are inherent to the paper-based prototyping concept.

At the same time Blended Prototyping aims to overcome the limitations of the paper-based procedure, by supplying a mixed-fidelity prototyping approach, where prototypes are tested directly on the target device. To be valuable for the development of more complex prototypes as well, the approach furthermore provides technologies to blend the paradigms of the throwaway and evolutionary prototyping processes by supporting the extension of prototype functionality with native programming code.

This motivates the third research question:

***Research Question 3*** *(RQ 3):*

Is it possible to create a new prototyping approach for mobile UIs, which adapts the full advantages of the paper-based prototyping approach, and at the same time adapts advantages of high-fidelity prototyping approaches to produce mixed-fidelity prototypes? Based on the discussion on prototyping goals above, such a system should meet the following motives:

- create a platform for interdisciplinary teamwork
- provide an approach that supports creative work
- provide an approach that is easy to learn
- facilitate the advantages of the physical use of pen and paper
- facilitate the advantages of the abstraction that are inherent to paper sketches
- deliver quick prototype results
- blend the paradigms of throwaway and evolutionary prototyping, and allow reusable programming of extended prototype functionality
- support on-device tests to allow even large-scale user tests in the real use context that take into account device related usability problems

The motives for the design of a new prototyping approach postulated above are derived from the discussion about shortcomings of current prototyping approaches held above in Sect. 2.3.4. The answer to that third research question is given in Sect. 4.4, after the description of the development and design of the Blended Prototyping approach.

Followed by that, the success of the Blended Prototyping to improve the prototyping process is evaluated. For this, the most relevant requirements identified in the catalog are used, to develop metrics for a comparative evaluation of the Blended Prototyping approach with two well-used prototyping alternatives. This investigation is motivated by my fourth research question:

***Research Question 4*** *(RQ 4):*

Is the newly developed Blended Prototyping approach able to improve the prototyping process, with regard to the previously identified requirements catalog?

This fourth research question is addressed in the comparative evaluation of Blended Prototyping that is described in Chap. 5.

# Chapter 3
# Prototyping Requirements

A thorough requirements analysis is the first step of a successful development. Here, principle design objectives are gained that steer the development efforts into the right direction. This being said, finding the most relevant requirements for a development tool is a complex task. Requirements can be gathered from different sources: Customers ask for specific demands, market studies can help to provide insights, and experience in the development team contributes to the analysis. Depending on the product, legal regulations, industry standards, and design guidelines can provide clarification in more detail.

A methodical approach to generate requirements for development tools that address the design of mobile UIs is less obvious. As touched before in Chap. 2, various tools exist that all aim to improve the prototyping process of mobile UIs. However, the requirements they point out to be relevant differ from one approach to another. A search for industry standards and documented best practices did not bring applicable results. Nor was the investigation of related scientific work able to provide a clear picture. Authors usually discuss requirements that are specifically related to their personal focus on the topic. A comprehensive taxonomy cannot be found.

Therefore I decided to address this gap by identifying a set of requirements from a literature review that was then evaluated in a survey with expert practitioners. As applied prototyping techniques usually change in the course of the project, I expected the requirements for prototyping tools to be different in single project phases as well. For this reason, the requirements were rated with respect to their importance at different product development phases.

This chapter describes the findings of my search for a set of requirements for mobile UI prototyping tools, which are rated by domain experts in their importance for different development stages. Such a catalog cannot only serve as a helpful design guideline for tool developers, but can moreover help to identify suitable metrics for the evaluation and comparison of different prototyping approaches. The chapter documents the identification and description of relevant requirements from



B. Bähr, *Prototyping of User Interfaces for Mobile Applications*,
T-Labs Series in Telecommunication Services,
DOI 10.1007/978-3-319-53210-3_3

a literature study. It then describes how these results were evaluated with expert practitioners that work in mobile app design and development projects.

## 3.1 Identification of Requirements from Literature Research

As a first step in developing a requirements catalog for mobile user interface prototyping tools a literature research was conducted, where related articles in the field of human computer interaction of the last ten years were systematically surveyed. The result of these efforts is summarized in Table 3.1, which lists the requirement categories with a short description and exemplary references.

The table provides a short description for each of the identified categories. A detailed description of each of the requirements is given in the following pages. Grouped into three segments, the table points out *requirements regarding the prototype design process*, *requirements regarding the prototype evaluation*, and *requirements affecting the prototype's nature*.

Related work draws a fragmented picture on requirements of tools for the prototyping of mobile app user interfaces. A comprehensive catalog addressing the topic cannot be found. Oftentimes, authors who present newly developed prototyping tools touch the topic. However, they usually do not refer to or present a holistic catalog, but present a small number of requirements that they take as granted without further discussion.

In the search for a comprehensive set of requirements, the last ten year's publications of the most relevant conferences to the topic were systematically analyzed and relevant references were followed. For this, the proceedings of the conferences CHI, Mobile HCI, CSCW, UIST, EICS, and HCI-Int. were regarded. Not only references that were targeted at mobile app prototyping were surveyed, but also texts that generally address the topic of prototyping were considered to draw general requirement conclusions.

The analysis resulted in a catalog of the 17 requirements that are described in the following. Exemplar references are included into these descriptions, which document the relevance of the single requirements in related work.

### *3.1.1 Requirements Regarding the Prototype Design Process*

#### *Freedom of Creativity and Design*

The tool allows designing in a free manner. Questions on accuracy can be postponed. Free creative work is promoted.

As Dolenc and Mäkelä [2] put it, we "still today […] need a system in which *freedom of creativity and design* is the primary concern". Being free in the design

**Table 3.1** List of requirements from the literature study

| Requirement | Description | Exemplar references |
|---|---|---|
| *Requirements regarding the prototype design process:* | | |
| Freedom of creativity and design | The tool allows designing in a free manner. Questions on accuracy can be postponed. Free creative work is promoted | [40, 127, 132] |
| Getting quick prototypes | The tool is targeted at delivering applicable prototypes as quickly as possible | [34, 86, 94, 145] |
| Independent parallel development of designs | The tool helps and motivates users, to work on design alternatives and consider them in the testing | [34, 70, 140] |
| Collocated group work | The tool is well suited to support groups to work simultaneously at the same place | [62, 91, 94] |
| Remote group work | The tool is well suited to support groups to work simultaneously at different places | [26, 65, 69] |
| *Requirements regarding the prototype evaluation:* | | |
| Support of expert reviews | The tool is well suited to conduct expert reviews | [34, 86] |
| Support of design reviews | The tool is well suited to conduct design reviews | [83, 111] |
| Tests in the real use contexts | The tool produces prototypes that can be tested in the same use contexts, as the later product | [40, 106, 127] |
| Easy setup and distribution of user tests | The tool supports in the setup and execution of user tests and supplies mechanisms to deliver the prototype to the test users | [62, 95, 106] |
| Simultaneous tests of different ideas | The tool allows a comparative test of different design ideas | [62,95,106] |
| Tests with a large number of users | The tool allows tests with large numbers of users, since test users can do the testing autonomously | [41, 106] |
| Automated model based evaluations | The tool supplies mechanisms, which can be used for automated model based evaluations of the interface prototype | [5, 72, 115] |
| *Requirements affecting the prototype' nature:* | | |
| Use of animations | The tool allows the use and definition of dynamic content and animations in the prototype test | [81, 83] |
| Advanced functionality of a prototype | The tool allows to produce prototypes of an elaborated functionality | [81, 96, 167] |
| Reusable prototypes | The tool makes it possible to reuse the prototype itself at later stages | [62, 95, 106] |
| Reusable programming code | The tool uses programming code that can be reused later in the development | [83, 95, 106] |
| Tests on different platforms | The tool creates prototypes that can be tested across different platforms, without major adaptions | [39, 139] |

processes promotes creative work especially in early stages, where the reconsideration and refinement of ideas is in the focus of the design. The *freedom of creativity and design* is therefore a requirement that is related to the ability of a tool to allow its users to operate freely in their creative processes. This means that the system does not distract users in the course of their ideation and design processes, e.g. by forcing them to interact with the system in lengthy dialogs. Users should be free to decide for themselves, on what time in their creative process they use which level of accuracy [29, 37].

Coyette [37] revealed in an experimental study that designers highly value a freedom of design that puts them in a position to be able to smoothly progress from simpler UI designs to versions of a more advanced level of detail. In other words, designers should be free to choose the level of fidelity in which they find and elaborate their UI design. For example, many designers consider hand sketches on paper as a most effective way to work on first drafts of future UIs [4, 38, 89].

#### *Getting Quick Prototypes*

The tool is targeted at delivering testable prototypes as quickly as possible.

The ability of a tool to deliver quick prototype results is essential for its applicability, especially in early design stages, where "prototypes should be lightweight and fast to create [… and therefore …] facilitate quick turnaround" [163]. In their first drafts, interface ideas are uncertain and need quick verification. The speed of a tool to deliver fast testable results is therefore important to avoid lengthy developments that might later turn out to be rejected.

In recent years, under the label *Rapid Prototyping* a big field of approaches has developed that considers speed as one of its most distinguishing features [2]. They speed up the prototyping of interfaces by reducing the aspects that are necessary for the interface definition.

#### *Independent Parallel Development of Designs/Simultaneous Tests of Different Ideas*

The tool helps and motivates users, to work on design alternatives and consider them in the testing. The tool allows a comparative test of different design ideas.

Especially in early design stages, app ideas are still vague and the best solution has still to be proven. In this situation, it is a promising approach to develop alternative ideas, which can then be tested in comparison to another. The more solutions are tested in parallel, the higher is the likelihood to in fact find the best suitable interface idea [29].

The ability of a prototyping approach to allow an independent parallel development of designs and their testing depends to a large extent on two of the requirements described above. A free creative work is necessary to find good alternative ideas; the ability to deliver fast prototypes is needed to keep the development of multiple ideas affordable. Most often, alternative ideas will regard specific aspects rather than the interface as a whole. For example, alternative ideas might exist about the structure of certain menus or the design of certain screens, whereas other parts of the app are already sufficiently validated. For the

development and the testing of different prototype variants it is therefore very helpful to have the opportunity to create variations of selected isolated parts of the app.

For the testing of prototypes that involve alternative designs, ways have to be implemented that allow the setup of different tests designs. This regards questions like, which kind of prototype variation should be addressed to what kind of test-user, is a within- or between- subject design more suitable, or should the user be enabled to choose between different interface alternatives.

***Collocated Group Work/Remote Group Work***

The tool is well suited to support groups to work simultaneously at the same place/at different places.

By far most software development projects are done in groups and not by isolated, independent computer-geeks. As a matter of fact, Sommerville [141] names communication skills and to be able to work collaboratively in groups as a key prerequisite for successful software developers. In most software development projects, ideas and decisions on software design and implementation issues are developed within groups. Therefore, development tools should support people working together on an idea [146].

Collaboration between group members can happen both, in person at the same location, or remotely at different locations. For either way of collaboration facilitating technique, it is a major challenge to support the communication and interaction between the members of the collaborating group. This does include verbal and gestural communication as well as considerations on how relevant content can be shared between the team members. Under the term CSCW (computer supported collaborative work) the matter is investigated in a big research field in the HCI community.

### *3.1.2 Requirements Regarding the Prototype Evaluation*

***Support of Expert Reviews***

The tool is well suited to conduct expert reviews.

Expert reviews are inspection methods, usually conducted by usability experts. They are analytic techniques that do not involve tests with users, which are comparatively lengthy and cost extensive. Hence, expert reviews are delivering faster results at lower costs as empirical methods like usability tests. For this reasons, the ability of development tools to support expert reviews is an important requirement [111].

The most prominent examples of expert reviews are heuristic evaluations and cognitive walkthroughs. The term heuristic evaluation was coined by Nielsen and Molich [111] and refers to the systematic review of a user interface by an expert

using a set of design guidelines. Such heuristics serve as a template to uncover problems users are likely to encounter.

Another important expert review method is the cognitive walkthrough, which was introduced by Lewis et al. [92]. In a first step, goals are defined users will likely have in the interaction with the software. Followed by that, the user interface is systematically analyzed for problems users might encounter following these goals.

***Support of Design Reviews***

The tool is well suited to conduct design reviews.

The design review, often referred to as software review, is a standard method in the process of software engineering. Design reviews serve primarily to control requirements related to the developed software, which were previously defined with the customer. They serve as a proof of concept and are used to identify the necessary technical means to successfully proceed with the development [23]. They have an important function as a communication tool between customer and software developer, informing the customer about the current status and development approaches, and giving the software developer the confidence to be on a development path that is approved and supported by the customer. Therefore prototyping approaches should produce prototypes that are applicable in design reviews.

In general, design reviews do not only help to discuss the user interface of software, but can ensure technical goals, and principles in the software design as well. As Blanchard [22] underlines, design reviews are a helpful tool at different design stages of a product. He categorizes the four different types of Design Reviews (DR) that are run from early to late phases of the software development lifecycle: Formal DR that is directed toward the final approval of a design configuration, Conceptual DR that is concluding the conceptual design phase, System DR where first results are checked, and Critical DR that builds the last evaluation milestone to evaluate the success of the project to meet the requirements.

***Easy Setup and Distribution of User Tests***

The tool supports the setup and execution of user tests and supplies mechanism to deliver the prototype to the test users.

After the design of a prototype is finished and its function is described, steps have to be taken to setup, distribute and conduct user tests. Whether or not these steps are easily and quickly achieved, depends on the nature and implemented methods of a prototyping approach.

Leichtenstern and André [91] underline that a prototyping tool should make it easy to set up and distribute user tests, by providing a strong link between its design, evaluation, and analysis components. A big difference in this respect exists between high-fidelity and low-fidelity prototyping approaches [66]. High-fidelity prototyping approaches usually require higher efforts to produce a runnable version of a prototype, especially since its functional aspects have to be described entirely. In comparison to that, low-fidelity prototyping approaches require a minimal effort

to produce a testable prototype version, but need substantial work and manpower in the testing.

In early development stages, where user tests are conducted with a limited number of users and the extent of the prototypes are limited, low-fidelity methods play out their strengths in delivering fast testable results. However, the higher the number of necessary test users, and the more complex the tested prototypes grow, the more urgent methods for an easy setup and distribution of user tests become [98].

***Tests in the Real Use Contexts***

The tool produces prototypes that can be tested in the same use contexts, as the later product.

As explained in detail above in Chap. 2, in comparison to stationary software, user interfaces of mobile applications have to deal with a number of specific challenges. Though the size of mobile devices seems to grow each year, they still hold a number of hardware limitations. To name a few examples, their network connection is not stable, their input precision varies, and problems like occlusion have to be dealt with. Beside that, the interface design of a mobile app has to take into account a number of different use contexts. Unlike software for stationary use, mobile apps are not just handled sitting at a desk, but walking on the street, driving in the subway, standing in the elevator, and in a myriad of other usage scenarios. Different use contexts bring different specific challenges in the device user interaction [127]. Factors that vary hugely lay for example in a changing user input precision, changing lighting conditions, and different forms of distraction.

Mobile apps very often provide services that are used oftentimes a day, but for a short time. The quality of such services cannot be simulated in a laboratory setup, but has to be experienced in the real daily use.

For these reasons, prototype tests of mobile UIs in the real use contexts are oftentimes required [66, 91]. Whether or not an app should be tested in the wild, depends on its nature and its development stage. The ability of a prototyping tool to allow mobile testing can be crucial for a development project [106].

***Tests with a Large Number of Users***

The tool allows tests with large numbers of users, since test users can do the testing autonomously.

Depending on the questions that need to be clarified by a prototype, the necessary number of test users varies. Especially when users are tested segmented in groups, higher numbers of subjects are necessary to measure statistically significant results. Such tests are particularly relevant in later development stages [41, 106].

Therefore, prototyping tools that address later development stages should provide mechanisms to conduct tests that include large user numbers with a reasonably low effort. Prototypes for such tests will be usually run directly on the target devices and processes for large scale study designs, realizations and analyses are implemented.

#### *Automated Model Based Evaluations*

The tool supplies mechanisms, which can be used for automated model based evaluations of the interface prototype.

In model based evaluation methods cognitive architectures like GOMS [32] or ACT-R [103] are used as models to predict the performance in human computer interaction. User models are applied to adapt computing systems to the specific user needs in a way, that they are able to "say the *right* thing at the *right* time in the *right* way" [51].

Model-based evaluations are to be used before user tests, and can already help to uncover and solve flaws in the implementation of user interaction. They cannot make user tests redundant, but can help to reduce the number of iterations for lengthy and expensive user tests. Therefore, automated model based evaluation can be a valuable tool in the development of mobile user interfaces [76].

### *3.1.3 Requirements Tool Implications on the Prototype's Nature*

#### *Use of Animations*

The tool allows the use and definition of dynamic content and animations in the prototype test.

Animations and dynamic visual components can be important for the user interface design, not only for the development of mobile games. Such content does not only play an important role for the outward appearance of the interface, but for its user interaction as well [74, 157]. Hence, their presence in prototypes should be considered. In current standard mobile development techniques, like side- or action-bars, can be used as a standard implementation from code. More customized animations have to be designed and programmed in a lengthy and complicated manner [156, 157].

For the development of software for stationary use, different approaches like the Adobe Director or animation capabilities in Microsoft Silverlight have been established, with which animations can be easier created and integrated to the software development process. Some of these techniques have been adapted to mobile apps [74, 157] that aim to facilitate the advantages of mobile apps in the animation. However, current standard development tools for mobile software still lack such capabilities.

#### *Advanced Functionality of a Prototype*

The tool allows producing prototypes of an elaborated functionality.

Prototyping is not only important in early design stages, but keeps its relevance throughout the whole development process [109, 138]. In early and middle design stages mockups can improve test results with low functionality and a materiality

different to the later product [44, 97, 140]. However, in late phases, where the prototype development is determined by pre-product versions, techniques have to be applied that allow for a richer and advanced functionality of a prototype.

Such advanced prototype functionality exceeds the definition of advanced visual content and includes any form of algorithm that enhances the dynamic behavior of a prototype. Here, to be able to define advanced functionality with the same technical resources and programing libraries as in the final product development is a clear advantage. This way, the full capabilities of the applied technologies can be explored and facilitated. Naturally, prototypes of an advanced functionality require a high development effort. Therefore, the reusability of the prototype, and especially of its programming code, is very important for an efficient development process.

***Reusable Prototypes and Reusable Programming Code***

The tool allows to create prototypes/uses programming code that can be reused later in the development.

As explained above in Chap. 2, the reusability of prototypes is a central criterion in the distinction of prototyping approaches. Where *throwaway* prototyping approaches relinquish on using reusable prototypes to achieve faster testable results, *Evolutionary* prototyping approaches employ reusable prototypes, which save development efforts in allowing to alter and build up to an existing prototype, rather than starting from scratch in each new iteration cycle.

Reusable prototypes grow in relevance in later development stages, where the effort of reproducing a prototype becomes more and more extensive.

Prototyping tools, especially if applied in later development stages, should therefore put developers into a position, where they can easily reuse their previous prototypes in a new iteration cycle [163]. This regards the prototype design, its programming code, as well as the technical infrastructure, in which it is embedded [37, 38, 49].

***Tests on Different Platforms***

The tool creates prototypes that can be tested across different platforms, without major adaptions.

Most mobile app development projects aim to address more than one target platform with their product. Usually apps are made available at least for the two most popular mobile operating systems, Android and iOS. For this reason, there is a need for prototyping techniques to provide prototypes that can be tested across different platforms [37, 38, 93].

For low-fidelity prototypes, which are not bound in their testing to a real mobile device, this requirement might seem unimportant. However, it should be remarked, that different mobile operating systems follow different user interface design and interaction concepts. Hence, even in low-fidelity prototypes, it is advisable to adjust the interface designs to different operating system concepts.

In high-fidelity approaches, technical requirements have to be solved to make the prototype executable on the same hardware as the later product. Many technologies generate prototypes that can be executed as web-apps, universally applicable on all mobile platforms. Others use so-called pseudo-code that is run inside a container app, which itself is compiled for different operating systems.

In the next step, in order to judge the importance of the single requirements at different stages of the development process, the presented requirements list was evaluated with experts. This evaluation and its results are described in the following two sections.

## 3.2 Assessment of the Requirement with Expert Practitioners

### *3.2.1 Study Objectives*

The experts that participated in the study brought a new viewing angle into the requirements analysis that me and probably most authors from scientific related work were lacking: distinct practical experience. In the study, the requirements catalog generated form literature research was meant to benefit from this experience in two different ways.

First, the experts were asked to suggest new requirements that might have been missed before. Second, the experts were requested to rate all single requirements with respect to their relevance for different development stages, as well as in their general importance throughout the whole product development process. This data was later used to identify the most important requirements for early, middle, and late development stages.

The suitability of a requirement for the desired catalog can be determined from two characteristics. First, a requirement is obviously relevant if it is ranked generally high throughout all phases of development. Second, a requirement might be ranked comparably low in the average over all phases, but at the same time is rated highly at a specific stage. In that case the aspect remains very relevant for the catalog to provide an orientation set for those tool developers, who target their prototyping mechanisms at particular development stages.

At the time of the expert evaluation, the Blended Prototyping platform (described in detail in Chap. 4) was already implemented in a first version, which the experts could use to try out and explore the technique on their own. Besides being asked about the general assessment of requirement categories, the experts were therefore introduced into the technology, to get familiar with the tool, and finally to give feedback on the approach. This feedback was given in two ways, first in an open discussion, and second in a rating of Blended Prototyping in regard to the requirements that were previously identified. This way, not only valuable feedback for the further development of the system was generated, but furthermore

the applicability of the catalog was tested, to be used as a basis to judge prototyping techniques. These study objectives can be translated into the following research questions:

RQ 1: Which of the suggested categories are generally most important?
RQ 2: What other requirements will experts name?
RQ 3a: Does the relevance of requirements change in different project stages?
RQ 3b: If the relevance changes over time, which requirements are specifically important in early, middle, and late development phases?
RQ 4: How is the Blended prototyping system judged in general and what are its biggest advantages and disadvantages?
RQ 5: How easy is it for experts to apply the suggested requirement dimensions in their rating, and how is the Blended Prototyping system related with respect to the suggested requirements categories?

### *3.2.2 Study Design*

(1) ***Participants***

A total number of 15 experts participated in the survey (5 female, 10 male). The experts worked for companies of different sizes, situated in Berlin and Potsdam. Potential candidates were contacted before the study to answer some general questions about their person and level of experience. As a prerequisite for the participation, only experts were invited that worked professionally in app development and design projects for at least two years.

Upon being asked to name their responsibility in the development process, the participants named roles that varied between *software developer*, *designer*, *user researcher*, and *project manager*. Maybe unsurprisingly for the Berlin startup scene, about a half of the responses included more than one role at a time.

As an indicator for the size of the development projects the participants are involved in, moreover, they were asked to give an estimation of the average number of users that used their apps. This number varied heavily in the given answers.

Finding experts for the survey was not easy. Beside the search to get in contact with suitable candidates, it was hard to motivate them to a degree that they were willing to spend at least two hours after their office hours in a survey. Hardly being motivated by financial compensations that are affordable to a university project, candidates had to be motivated by creating interest for the conducted research.

Therefore in preparation I talked to candidates on the phone to introduce them to the general purpose of my research. I mentioned the existence of a new self-developed prototyping tool, which participants would get to know and be asked feedback about. Candidates interested in participation were asked not to do research on my previous work, but to participate in the survey with no specific preparation.

(2) ***Procedure and Collected Data***

For the expert talks two interview methods were considered: a group discussion, and the personal talk with one expert at a time. Group discussion can develop a positive momentum, were the participants are inspired by each other in their contributions, but at the same risk that single participants are holding back their thoughts. I wanted to learn as much about the personal opinion and experience of the experts as possible, hence I invited them to participated in the survey one at a time. At the start of each session, the general purpose of the study was explained and permissions were asked for, to record the necessary video and questionnaire data. The survey was structured into three parts that are described below. All sessions were recorded on video for a later analysis of the demo sessions and talks with the participants. All questionnaires were answered within a computer survey called *Limesurvey* [171]. The tool allows the implementation of custom scripts, which were used to dynamically refer some questions to answers the participants gave previously. At all phases of the survey the experts were free to ask content-related questions.

*Pre-questionnaire*

A first computer supported pre-questionnaire started with questions on demographics and personal experience in app development and design. Participants were then asked to write short open texts on how their usual development processes are structured and which development tools are used at what stages. Followed by that, the experts were introduced to the requirements that had been identified from related work.

Now, the participants were asked to rate each requirement with regard to its importance (5 point scale, ranging from 1, *extremely unimportant* to 5, *extremely important*) at five different project stages (ranging from *very early* to *very late*). Answers were given in a matrix view, containing five data points for each requirement.

In the following, the experts were asked to suggest additional requirements that might have been missed in the presented catalog. The next page of the survey then contained a dynamically generated question, in which the participant was asked to rate the importance of the requirements she suggested in the same matrix style as before.

*Demo and Free Discussion*

On completing the pre-questionnaire, the participant was guided to a separate room, where the Blended Prototyping platform was set up. Upon an explanation of the fundamental ideas of the approach, the expert was guided through a participatory demo on the system usage. The demo followed a fixed step-by-step script, allowing the experts to control the setup themselves. The experts were free to try out the system in whatever way they felt like. They concentrated on exploring the system functionality in setting up small prototypes that they quickly invoked on mobile devices.

Questions and debates about the system were allowed and promoted at any stage. The demo led to a free discussion, where general feedback on the approach and its implementation was provided, its applicability to existing prototyping processes was reflected, and the experts' opinion on the most important next steps for the Blended Prototyping approach was given. To allow later analysis, the sessions were recorded in video and audio.

*Post-questionnaire*

After the debate following the demo session came to an end, a second computer questionnaire was presented to the experts. Here, the experts were asked to rate the Blended Prototyping platform on the basis previously identified requirements, both from literature and from the suggestions of the individual participant. This was done in a matrix, were the capability of the prototyping approach was to be rated to meet a certain requirement (values ranging from 1, not at all supported, to 9, very well supported) at five different development stages (5 point scale, ranging from *very early* to *very late*). Finally, the participants were asked to give free feedback in open text fields.

(3) ***State of the Design Tool***

The Blended Prototyping platform has been substantially developed further after the expert reviews. Hence, the platform as described in the following chapter differs in various aspects to the one used in the two last steps of the expert review. The main reason for presenting a preliminary version of the platform to the experts was that I wanted to test it as early as possible, to gain feedback for the further system development. The implementation of a number of low-tech manipulation tools, of advances in the tablet control application, and a better performing internal object communication were implemented after the study. However, the platform was at a state, in which all concepts of the approach became visible.

### *3.2.3 Results of Expert Survey*

In the following, the results of the expert survey are displayed, structured according to the research questions formulated above. The most important results were published before in a HCI-International conference paper [10].

RQ 1: *Which of the suggested categories are generally most important?*

The experts were specifically asked about the importance of the suggested requirement dimensions at different design stages. As a measure for the general importance of a category, its average rating throughout these stages is used. In Table 3.2, shows the requirements sorted by to their average ratings. The table is segmented into two areas: the upper area shows all categories that were identified from literature, the lower area shows the ones newly suggested by the experts. Only

**Table 3.2** Generally most important requirements (due to their lower n, means for suggested categories are marked with an asterisk)

| Requirement | Mean |
|---|---|
| Collocated group work | 4.1 |
| Getting quick prototypes | 3.8 |
| Reusable programming code | 3.8 |
| Freedom of creativity and design | 3.6 |
| Tests in the real use contexts | 3.6 |
| Support of design reviews | 3.6 |
| Support of expert reviews | 3.5 |
| Easy setup and distribution of user tests | 3.3 |
| Simultaneous tests of different ideas | 3.2 |
| Tests on different platforms | 3.2 |
| Reusable prototypes | 3.2 |
| Use of animations | 3.2 |
| Tests with a large number of users | 3.0 |
| Advanced functionality of a prototype | 3.0 |
| Remote group work | 2.9 |
| Independent parallel development of designs | 2.7 |
| Automated model based evaluations | 2.0 |
| (Suggested; n = 2) usability of the tools themselves | 5* |
| (Suggested; n = 1) fun to use the tool | 5* |
| (Suggested; n = 1) compatibility of the tool with different platforms | No rating |
| (Suggested; n = 1) open source availability | No rating |
| (Suggested; n = 1) tutorials and help for the tool | No rating |

those experts who themselves suggested the new requirements gave them a rating. Therefore, the number of raters who contributed to the average rating of those dimensions is very small. Moreover, due to the fact that an expert would only suggest a new dimension that she thinks is important, the newly suggested categories are naturally rated comparatively high. Hence, the ratings for the new dimensions cannot be directly compared to those of the ones suggested from literature.

The categories ranked highest were *collocated group work* (4.1), *getting quick prototypes* (3.8), and *reusable programming cod*e (3.8). The three lowest rated categories are *remote group works* (2.9), *independent parallel development of designs* (2.7), and *automated model-based evaluations* (2.0).

Due to their ranking below of the middle point of the scale at 3.0, the three lowest categories might be considered rather unimportant. However, this does not necessarily mean, that they should be discarded; a closer look at their rating the single design stages is necessary. A requirement ranked low in average can still be highly relevant in specific phases and should therefore be preserved in the catalog.

As displayed in the figures on the following pages, means for *remote group works*, ranging from very early to very late, are (3.2, 3.2, 3.0, 2.8, 2.4), for

*independent parallel development of designs* (3.1, 3.0, 2.5, 2.4, 2.3), and for *automated model-based evaluations* (2.2, 2.0, 2.2, 2.1, 1.5). The sequences show, that the first two categories are more important in the beginning of the development process, where they are rated above scale's middle value of 3.0. For this reason they should be considered in the catalog.

For the *automated model-based evaluations* category however, the ratings are in no phase higher than 2.2. Therefore I decided to discard this category from the further discussion.

RQ 2: *What other requirements will experts name?*

The experts suggested a very limited number of new requirements. Only 3 of the 15 participants gave new recommendations and no more than 5 new requirements were proposed. *Usability of the tool itself* was the only suggestion that was named by two different experts. The others, *tutorials and help for the tool*, *fun to use the tool*,

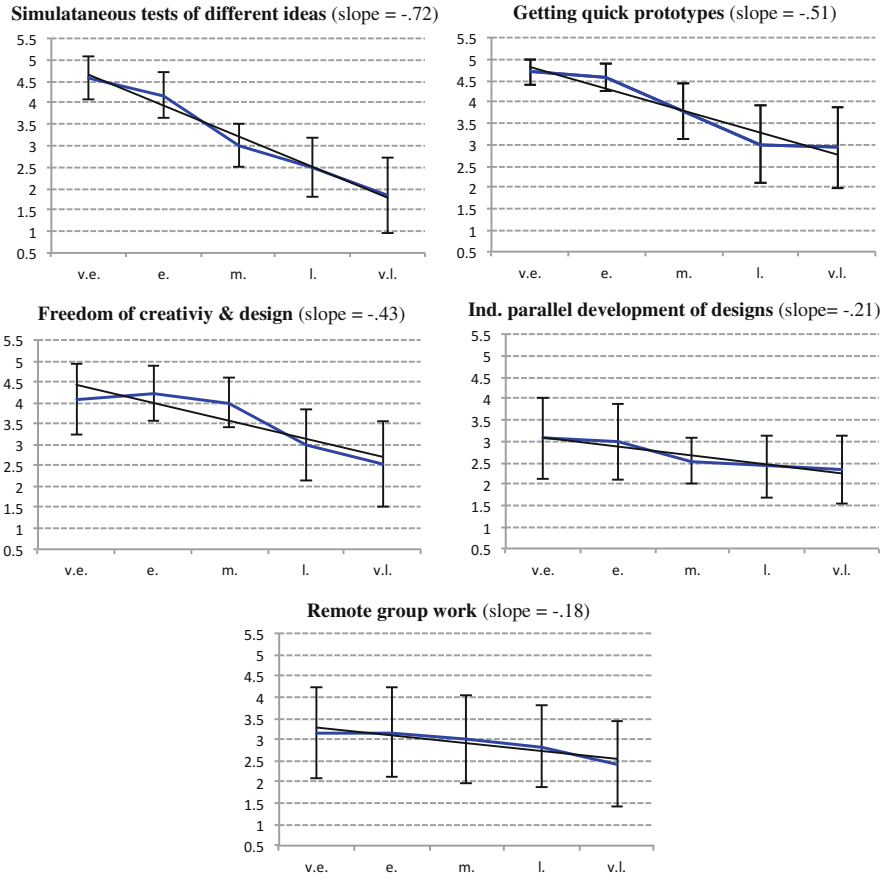


**Fig. 3.1** Requirements with increased importance at early stages. (Error-bars ± Std. Error)

*compatibility of the tool with different platforms*, and *open source availability of the tool* were just named once.

The ratings for the newly suggested categories can be found in Table 3.2. As discussed above, due to the limited number of their raters and due to a positive personal bias towards own suggestions, the ratings for the new items cannot be compared directly to the ones identified from literature review.

RQ 3a: *Does the relevance of requirements change in different project stages?*

The results show that the importance of the requirements varies over time in different ways: some requirements lose in importance over time (see Fig. 3.1), others are equally important throughout the process (see Fig. 3.2), and yet others become more important towards the end of the development (see Fig. 3.3). The difference between lowest and highest value is most prominent for the dimensions of the *simultaneous tests of different ideas* (Δ2.8), *tests with large numbers of users* (Δ2.5), and *tests on different platforms* (Δ2.3).

As a metric to group the categories accordingly, a linear regression for the development of each category was calculated. The categories were then sorted in regard to the slope coefficient of their linear regression.

The slopes calculated in the linear regressions are statistically significant ($p < 0.05$) for all but the three categories marked with an asterisk in the following figures. This concerns two requirements that are categorized to be stable over time: *support of expert reviews* ($p = 0.454$) and *reusable prototypes* ($p = 0.275$). For the

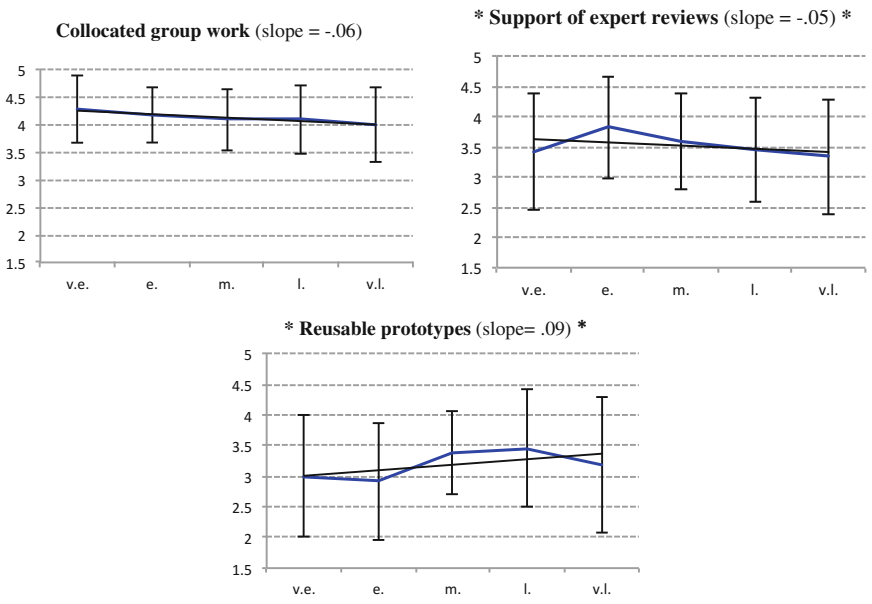


**Fig. 3.2** Requirements comparatively stable over time *non-significant slopes marked with asterisks*. (Error-bars ± Std. Error)

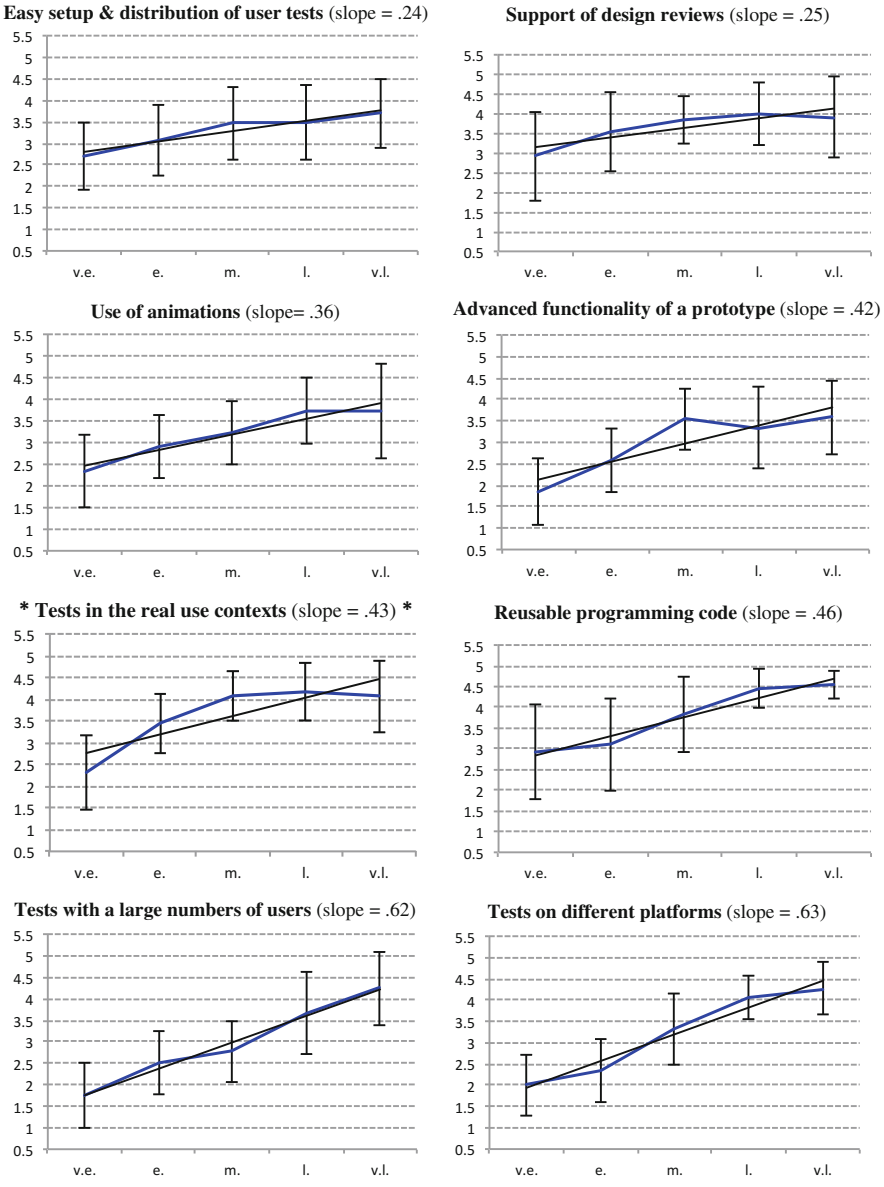


**Fig. 3.3** Requirements with increased importance at later stages *non-significant slopes marked with asterisks*. (Error-bars ± Std. Error)

**Table 3.3** Sorted lists of requirements for 'earlier', 'middle', and 'later' phases

| Earlier phases | | Middle phases | | Later phases | |
|---|---|---|---|---|---|
| Getting quick prototypes | 4.6 | Collocated group work | 4.1 | Reusable programming code | 4.5 |
| Simultaneous tests of different ideas | 4.4 | Tests in the real use contexts | 4.1 | Tests on different platforms | 4.2 |
| Collocated group work | 4.2 | Freedom of creativity & design | 4.0 | Tests in the real use contexts | 4.1 |
| Freedom of creativity & design | 4.2 | Reusable programming code | 3.8 | Collocated group work | 4.1 |
| Support of expert reviews | 3.6 | Support of design reviews | 3.8 | Support of design reviews | 4.0 |
| Support of design reviews | 3.2 | Getting quick prototypes | 3.8 | Tests with a large number of users | 4.0 |
| Remote group work | 3.2 | Support of expert reviews | 3.6 | Use of animations | 3.7 |
| Independent parallel development of designs | 3.0 | Advanced functionality of a prototype | 3.5 | Easy setup and distribution of user tests | 3.6 |
| Reusable programming code | 3.0 | Easy setup and distribution of user tests | 3.5 | Advanced functionality of a prototype | 3.5 |
| Reusable prototypes | 3.0 | Reusable prototypes | 3.4 | Support of expert reviews | 3.4 |
| Easy setup and distribution of user tests | 2.9 | Tests on different platforms | 3.3 | Reusable prototypes | 3.3 |
| Tests in the real use contexts | 2.9 | Use of animations | 3.2 | Getting quick prototypes | 3.0 |
| Use of animations | 2.6 | Simultaneous tests of different ideas | 3.0 | Freedom of creativity & design | 2.8 |
| Advanced functionality of a prototype | 2.2 | Remote group work | 3.0 | Remote group work | 2.6 |
| Tests on different platforms | 2.2 | Tests with a large number of users | 2.8 | Independent parallel development of designs | 2.4 |
| Tests with a large number of users | 2.1 | Independent parallel development of designs | 2.5 | Simultaneous tests of different ideas | 2.2 |
| Automated model-based evaluations | 2.1 | Automated model-based evaluations | 2.2 | Automated model-based evaluations | 1.8 |

category *tests in the real use context* the significance of the regression analysis almost approached significance ($p = 0.065$). For all other dimension, a statistically significant linear dependency of the requirements' importance on the different development stages was found.

RQ 3b: *If relevance changes over time, which requirements are specifically important in early, middle, and late development phases?*

Table 3.3 shows the requirements, sorted in three different ways. To present the results more clearly, the first two development stages, as well as last two ones, were aggregated to the scales *earlier phases* and *later phases*.

Very highly ranked for earlier stages are the categories *getting quick prototypes*, *simultaneous tests of different ideas*, and *freedom of creativity and design*. In contrast to that, the categories *reusable programming code*, *tests on different platforms*, and *tests in the real use context* are rated highest for the late development stages.

The list underlines once more the changing relevance of the attributes over time, as indicated with the different colored highlighting. The only category ranked in the top 4 categories in all development stages is *collocated group work*. In the mid-field of all stages are the rankings for the categories *design reviews* and *expert reviews.*

RQ 4: *How is the Blended Prototyping system judged in general and what are its biggest advantages and disadvantages?*

In general, the participating experts gave very positive feedback on the Blended Prototyping approach. They granted the review sessions a very considerable amount of time, especially in the concluding discussions about the new prototyping approach. As part of the post-survey, the experts were asked to describe the biggest advantages and disadvantages of the Blended Prototyping. Here, 13 of the 15 experts regarded the *participatory work in groups* and 10 saw the *deployment and testing directly on the mobile device* as a major advantage of the technique.

Beside that, 12 experts remarked the system's *modularity and its adaptability to different purposes*. About half of the experts regarded the system to be *easy to use* (7/15) and underlined the importance to *work with paper* (8/15). Furthermore, 6 experts wrote that the system is *fast*, 5 highlighted its ability to be *well suited for interdisciplinary work*, and 4 remarked that the system is *fancy* or that it would produce a *wow effect when used with a customer*.

Twice, experts stated that the system *motivates to produce reusable code from day one* and that its *hardware setup is inexpensive and is based on components that exist in most offices*. In opposition to that, six experts regarded the system setup as a big disadvantage, stating that *the setup is too complex*. Furthermore, it was stated by three experts that *coding is required too early*. Besides, two experts criticized the use of mobile devices in the design process and criticized that there would be *no easy definition of widgets by just using pen and paper*. As explained above, at the time the expert reviews took place, the low-tech tools of the design tool described in Chap. 4 were not yet implemented.

RQ 5: *How easy is it for experts to apply the suggested requirement dimensions in their ratings, and how is the Blended Prototyping system rated with respect to the suggested requirements categories?*

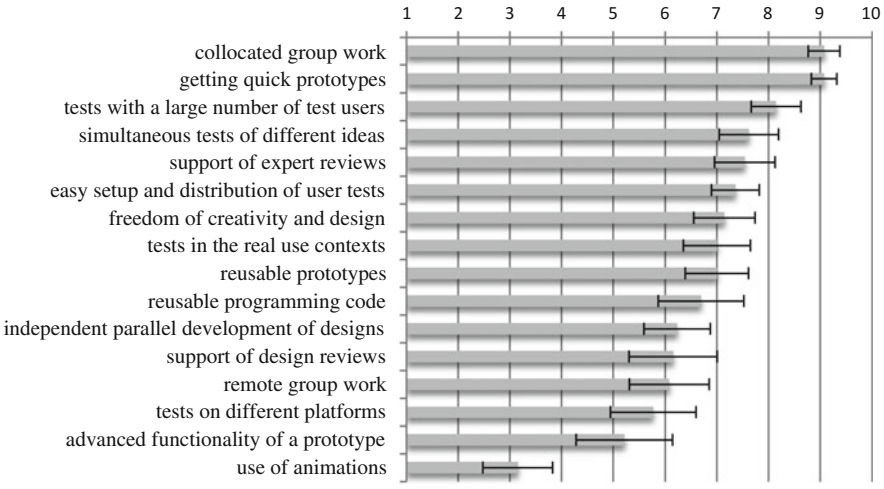


**Fig. 3.4** Expert ratings of blended prototyping with requirements identified from literature. (Error-bars ± Std. Error)

In the post-questionnaire the experts were asked to rate Blended Prototyping with the requirements catalog we identified before from literature, along with those categories the suggested in addition. Ratings were done on a 10-point scale, ranging from 1 = "really poorly suited to meet the requirement" to 10 = "really well suited to meet the requirement".

The experts had no difficulties in applying the categories. Questions were asked solely about the requirement 'automated model based evaluation', since no such features are implemented and the experts felt confused about how such automated evaluations might be included in the design tool. As shown in Fig. 3.4, Blended Prototyping's ability to support *collocated group work* (9.1) and to *get quick prototypes* (9.1) were rated best, followed by *tests with a large number of users* (8.1). The weakest ratings were given for the categories *tests on different platforms* (5.8), *advanced functionality of a prototype* (5.2), and the *use of animations* (3.2).

## 3.3 Discussion of the Results

The expert review was able to confirm 16 out of the 17 requirements found in the literature review as being important for development tools that address the design of mobile user interfaces. Since the requirement of *automated model-based evaluation* was rated very low both, generally as well as singularly for each development stage, it was discarded from the requirements collection. The number of additional new requirements suggested by the experts was limited. The newly suggested requirements were mostly related to the *usability of the tool itself* and its *joy of use*.

Of course, this does not prove that the created catalog is complete. In fact, an all-comprehensive catalog might be impossible to find. However, the fact that the experts did rarely name new requirements does indicate, that the results from literature did not miss too many obvious factors. This being said however, especially regarding requirements that may just be important at certain development phases, there will surely be aspects that might still be worth being added to the catalog.

Maybe, if asked before the presentation of the requirements set from literature, the experts had been more motivated to give suggestions. Maybe, the experts felt too tired after the somewhat lengthy rating of a set of 17 variables, each for five phases. Explanations are hard to find in retrospective. Possibly, asking the experts in a focus group would have improved the overall feedback. On the other hand, such open discussions bear the risk that single participants do not get involved in the discussion, because they feel too shy to hold up their personal opinion against others.

The experts had no issues in understanding the suggested categories and to rate their importance at different development stages. The results highlight the change of the importance of the individual requirements at different stages of the development process. They provide three sets of catalogs that collect the most important requirements for early, middle, and late design stages. Developers of prototyping

tools should carefully consider, which development phases they intend to address with their developments and use such information to concentrate the implementation of their approaches on the most important aspects. A successful Swiss-army-knife-approach, applicable in all development situations, does not seem to be feasible. The requirements catalogs cannot only be used as a reference for the development of new systems, in addition it can serve as a set of metrics for the evaluation of prototyping tools.

Besides being asked to rate and add to the requirements catalog, the experts were introduced to the Blended Prototyping approach and tried out its single modules themselves. This was done for two reasons: First, to check whether the experts had problems in applying the requirements to rate a prototyping tool. Second, the experts were introduced to the system to gain feedback on the strengths and weaknesses of the approach and to hear their professional opinion about the most important next development steps.

Generally, the experts gave very positive responses on the system. They saw a purpose and need for such a system and underlined its suitability for the collaborative work in creative tasks. The ratings of the system were very good, especially for the phases that the generated catalogs identified to be important for early and middle development phases, the quick testing of prototypes and the collocated group work.

However, it must be remarked, that the experts rated the system on a rather conceptual level. They did not actually use the system in a group to work in a productive task, but got to know the tool in hello-world examples. Of course, one could assume that the experts have a good idea on how the approach performs in actual usage. The results underlined a high potential of the system concept and its implementations, however, evaluating the system in real user tests are necessary to draw well-grounded conclusions.

As some experts found joy in tickling and tweaking the system, I was able to create a valuable list of bugs in the implementation. Beside that, as the result of the remarks by a number of experts, I decided to extend the design tool with further functions. This concerned the implementation of low-tech tools to avoid the use of the high-tech tablet devices in the design process (see Sect. 4.2.2), as well as the implementation of approaches for the prototyping of animations (see Sect. 6.2).

# Chapter 4
# Blended Prototyping—Design and Implementation

This chapter describes the idea and motivation of Blended Prototyping, displays its usage and discusses a selection of design decisions that were made in the system development. For this, in the first section of this chapter, the development of the approach is explained, where its principle design paradigms are displayed and the role of these paradigms in other prototyping approaches is discussed. Followed by that, the important role of feedback in the development of Blended Prototyping is underlined and the multiple occasions are highlighted, at which reactions towards the system could be gained in demonstrations, user tests, interviews, and discussions.

In the Sect. 4.2, the use of the Blended Prototyping environment is presented in its three modules. The text here concentrates on the perspective of the platform users and describes all the necessary aspects to understand how the approach is used in the design and development processes. The most important technical aspects of the platform implementation are then discussed in concluding Sect. 4.3.

## 4.1 Approach and Development

### *4.1.1 Blended Prototyping Design Paradigms*

As displayed in the early publications regarding the topic [11, 19], the primary goal of the concept of Blended Prototyping is to provide development teams with new processes and tools that can be used in earliest design stages already, to conduct a prototype driven development of mobile user interfaces. To get a better understanding on the specific needs of mobile developers and the conditions they usually have to cope with, I talked to different professionals from the Berlin startup scene when I planned and started to implement the first Blended Prototyping approaches. As a bottom line I learned that app development projects can only be successful, when they address a specific problem at the right time with a both smart and creative solution. In



B. Bähr, *Prototyping of User Interfaces for Mobile Applications*,
T-Labs Series in Telecommunication Services,
DOI 10.1007/978-3-319-53210-3_4

comparison to the world of desktop computer development projects, where big companies develop mainly large software solutions, mobile apps are usually created in businesses with very limited resources in small but effective teams. And though huge success stories are frequently told, like the ones of Instagram, Endnote, or WhatsApp, most app development project do not produce success stories.

Blended Prototyping was therefore designed to support app development teams in their collaborative creative thinking designing and developing mobile user interface prototypes. The approach aims to put designers in a position, where they are able to produce prototypes in a collaborative context as quickly and easily as possible, and that way, are able to progress their user interface ideas on the basis of iterative tests. At the same time, the approach should allow for the testing of prototypes directly on mobile devices.

As described detail above in Sect. 2.1.2, paper-based prototyping is an approach with very similar design objectives, but has its limitations when testing prototypes in the mobile context. For this reason Blended Prototyping aims to adapt the advantages of the traditional paper-based approach, but at the same time to deploy prototypes on mobile devices to enable easy tests in the wild.

As discussed in more detail in [13], Blended Prototyping focuses its design processes on the use of regular paper. When comparing the use of paper and pen to digital substitutions like stylus tablets, different authors [14, 20, 104] found that physical paper allows a faster and more natural interaction in teamwork and brings benefits in collaborative brainstorming processes.

To establish a seamless transformation of physical paper to digital prototype data, Blended Prototyping employs an overhead projected tabletop computing system, where physical and digital content on the paper are *blended*. Tabletop computing environments are frequently used to build collaborative systems with tangible objects, e.g. [142, 151, 165].

To not interfere with the creative teamwork, Blended Prototyping is designed to be as unobtrusive as possible. It can be used for designing and testing simple prototypes on the basis of paper sketches as fast as possible. However, additional programming code can be added to the prototypes, so that the complexity of their functions can be shaped as complex as needed.

### *4.1.2 Feedback Driven Development*

Blended Prototyping was developed in a number of iterations that improved the platform in different aspects. Compared to the initial version of the platform that was described in my diploma theses [7] and student research paper [8] the system was redeveloped in each component. One major change was the reconsolidation of the graphic framework used to display the content on the tabletop surface from the no longer supported and by now buggy Pidoco2D framework to JavaFX, which is now part of the standard Java SDK. Another important change was in implementing a different approach for the controls of the cameras. In earlier versions, this was

done with closed source development kits that were bound to Windows systems. Now, open source toolkits are used that are usable cross platform, provide a better image quality, better ways to maintain the programming code, and a better compatibility with a whole range of different cameras. Furthermore, the control of the tabletop-application was totally renewed, from initially being controlled by mouse input, to a system that uses a combination of physical tangible objects and an app that is ran on a tablet device. Moreover, a new object structure was implemented, the communication between servers and clients was shifted from http requests to full-duplex secure web-sockets.

Some of these changes derived from technical necessity, most others were triggered by practical application and evaluation of the prototyping platform. In the development of the Blended Prototyping platform, I aimed to follow the key idea of prototyping as well as possible, in iteratively progressing ideas on the basis of insights gained from tests with experts and users.

In the early conceptual stage, the development based on feedback that was given by colleagues, friends with app development and design background, and eventually fellow researchers I got to know at scientific conferences. Then, soon after the platform existed as a whole in its initial version, I used the tool two years in a row (2012, 2013) in the context of a workshop with 15 years old school kids, who participated in a one-week introduction to developing mobile apps. Other amateur user feedback was generated in participating at the "Lange Nach der Wissenschaften" in 2012, 2013, and 2014, where the Berlin regular public has the chance to get presented the work of researchers in different facilities throughout the city.

The approach was discussed with professionals from the research community in different formats. I addressed the communities of the conferences I visited, which laid in the MobileHCI (Int. Conf. on Mobile Human Computer Interaction), the CHI (Int. Conf. on Human Factors in Computing Systems), and the DSMB (Design Modeling Symposium Berlin). Beside that, Blended Prototyping was part of the interdisciplinary research project '*Rethinking Prototyping*' that was held as a cooperation between the *University of the Arts* and the *Technische Unitversität* in Berlin between 2012 and 2015.

Moreover, in 2013 the platform was used for half a year in a project conducted with the Deutsche Telkom AG, were it was used in four iterations to develop a prototype for a mobile app to facilitate shared work. Within this context, the platform was adapted and revised several times.

## 4.2 Blended Prototyping—System and Process Architecture

### *4.2.1 System Overview*

As illustrated in Fig. 4.1, the Blended Prototyping platform consists of three modules that each supports the prototyping process in different phases: from the

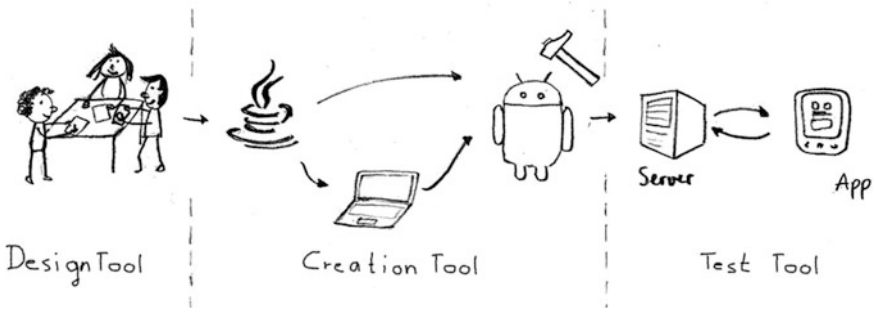


**Fig. 4.1** Overview—"blended prototyping" platform

initial design, the creation of the prototype, to the test of a prototype and the analysis of its results.

In the first module, the Blended Prototyping design tool, techniques are implemented that support groups of designers in the collaborative work on a prototype design. For this, a custom tabletop-computing environment was built, which allows designers to come together and discuss and progress their interface design ideas on the basis of simple paper sketches. The environment is equipped with an overhead video projector, which provides a bridge between the physical content on the table surface in form of paper sketches, and the digital content of the evolving mobile user interface. Within the sketching process, functional descriptions of the prototype can be defined, which are then highlighted on the table surface to provide a better overview of the user interface behavior. The design tool was topic of a publication at the Design Modeling Symposium 2013 [13].

The second module provides the necessary processes, to convert the data generated in the design tool into a prototype that is runnable on Android devices. Here, two alternative processes can be chosen: either a fully automated process, where testable prototype results are generated as quick as possible, or a manual process, where in a sub-step programming code is generated that can be edited by the development team to describe behavioral aspects of the prototype in more depth.

The third module includes a number of different tools that build an infrastructure for conducting and analyzing prototype tests, even with higher numbers of users. Here, a client-server solution for the setup and distribution of user tests is covered as well as techniques for the logging and analysis of usage data, created within the user tests.

The three modules are not necessarily used sequentially. When changes in previous phases have to be made, this can be done without restarting the whole process.

## 4.2.2 Module 1—The Design Tool

### 4.2.2.1 Tabletop Environment and Basic Design Process

Center of the Blended Prototyping design tool is a regular table, where a design team can meet to lay out and discuss their design ideas. The table is set up with technical equipment, which is able to transform the regular table into a tabletop computing environment. Specifically speaking, a tripod is positioned above the table surface, on which a video projector, a photo camera, and a video camera are installed (compare left photo in Fig. 4.2 on following page). These components are connected to a computer system, which employs the video projector as an output channel, and the two cameras as input channels for the interaction with the system's users.

The design tool is controlled either with a mobile app, which is installed on tablets that are supplied with the environment, or with the help of physical, tangible tools. Therefore, for each of the tool's system functions two alternative solutions are implemented. I implemented both techniques in the tool for the reason that they are complementing each other in their advantages and disadvantages. A detailed discussion of this is found below in Sect. 4.3.1 (3).

Designers use the table as meeting point to discuss their user interface ideas and to bring them, literarily speaking, onto paper. The design tool puts physical sheets of paper into the center of the design process. Just as in the regular PBP approach, designers use these paper sheets to draw their ideas, build collages, or apply whatever techniques they like. Barcodes printed on the paper sheets, allow the system to identify and locate the paper at all times (compare right photo in Fig. 4.2 on following page).

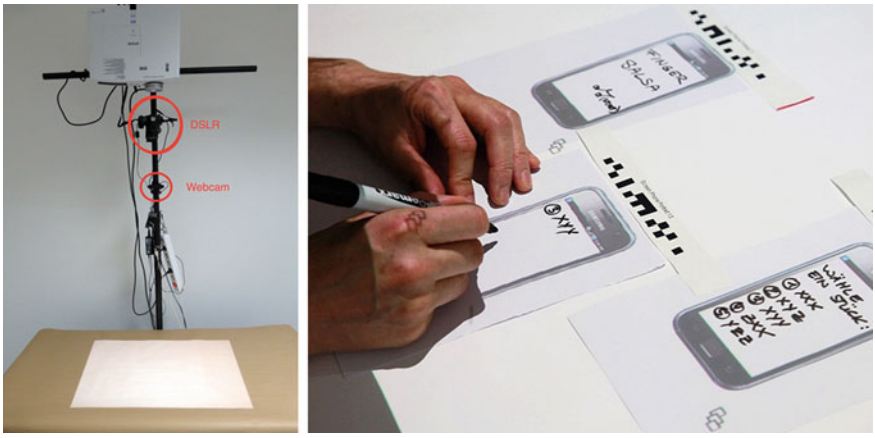


**Fig. 4.2** System overview (*left* hardware-setup, *right* hand-sketching in blended prototyping)

Whenever the designers are satisfied with a sketch, they can use the system to convert the physical paper sketch into a digital version. For this, the photo camera of the system is employed. Upon this digitization, the former physical content is now projected onto the table surface in a digital version. Whenever the designers feel like it, they can add additional sketches into the digital projections, which are subsequently added to the digital version as well (compare section below: *Process of sketching in a mixed physical and digital world*).

Within the sketches, the designers can define user controls that build the basis for the dynamic interaction of the user with the later prototype. A range of seven different user controls (find detailed description below) can be positioned and sized on top of the drawings. Depending on the control's nature, additional attributes can be defined. For example, links for buttons can be defined, that the interface should follow when the button is pressed. The defined user controls are projected as an overlay to the sketches on the tabletop surface. Some additional aspects of the user controls are highlighted on the surface as well. For example, linkage paths of buttons are displayed as lines connecting the button with the screen it leads to. This creates a storyboard view that provides a good overview of the developing user interface (compare section below: *Process of defining user controls and interface functionality*).

The design tool aims to be as unobtrusive to the design process as possible. Users are therefore free to decide at which time they want to use certain system functions, and at which time they prefer to ignore the system and rather concentrate on their design ideas. In the beginning of a design session, users will usually concentrate on their physical sketches, later on, when the interface prototype becomes more elaborated, the functions of the design tool will be addressed more frequently.

#### 4.2.2.2 Overview of the Key Interaction Techniques in a Design Session

***Role of the Barcode-Markers***

Paper sheets are identified on the tabletop surface with the help of printed barcode markers. The markers, initially introduced by Rohs et al. [84], are optimized to be recognized in their exact position and rotation in a camera image. This way, the computer system is always to recognize which paper sheets are present at which exact position on the table surface.

Different number spaces encoded in the barcodes represent different types of screens. Currently, screen designs for smartphone and tablet apps can be used, each either in portrait or landscape orientation (see Fig. 4.3). In addition to that, *zoom* markers are available for both smartphone screen types (the middle two screens in Fig. 4.3). These screens are views to the same content as its regular siblings, they do however display the content in a doubled size.

**Fig. 4.3** Different device types: smartphone, smartphone zoom, and tablet

The screens of larger size are used when more space for the drawing is needed, the smaller screens help to get a better idea of the actual proportions and look of the interface on the mobile device. Moreover, a larger number of small screens fit on the table surface at once. Hence, they are more useful to provide an overview of the user interface as a whole.

***Process of Sketching in a Mixed Physical and Digital World***

Drawn interface content that is displayed on the paper sheets can be in physical or digital form. Physical content is created with pen drawings on the paper, or similar techniques. Digital content is created in the conversion of physical content into the digital space.

The process of digitizing drawings is controlled by the development team and can be initiated at any suitable time. For this, either the design tool app, or the photo-trigger marker is used. Upon the digitization process is started, a photo of the whole tabletop is taken by the system's photo camera. Now an algorithm filters out and corrects the screen images the users wanted to photograph. Now, the paper sheets a photo has been made of are substituted with blank paper sheets that have the same printed identification marker. After this, the digital interface version is projected onto the paper, just where seconds ago the physical content was drawn.

Additional physical content can always be added to the projection of the digital screen version. In the next digitization step, this new content is isolated from the screen background and added as new layer to the existing interface sketch. This way, new drawings seamlessly become part of the evolving screen design. Having interface content as a digital version has benefits that are common in every image

processing software: Content can be copied between the screens, mistakes can be easily undone, and a history function allows browsing through and restoring previous screen versions. The process of digitization helps users to focus their work on paper sheets, where digital and physical interface content is merged. Users are free to choose when digitization steps should happen; therefore it is them who control which content should be kept at which design stages. Where the physical world helps the natural design of ideas, digital content brings its advantages reproduction and versioning.

The digitization process can be started either with the help of the table control app, or with the help of the physical take-picture tool. Selecting the camera icon in the app's top menu will start the digitization process. Now, two alternative views can be used to select the screens that the user wants to digitalize. In the first view, a live view of the tablet's rear camera is shown with a target area in the middle of the screen. By catching a screen marker in this area, the screen is selected to be photographed. To provide the user with a feedback about the selection, the tablet plays a beep and the selected screens are highlighted with a projection. Screens can be de-selected by repeating the scanning with the tablet.

Alternative to the scanning process, users can use a second view on the tablet, which allows them to manually select the screens that shall be photographed. Here all screens known by the system are shown in a list, where they are presented in form of a small image that shows their last digital design. Checkboxes on each of the displayed screens allow the user to select and deselect the screens for the photo process. The confirmation of the dialog triggers the camera, the sketches are isolated from the camera image, and the newly created digital sketch versions are now projected on top of the paper screens.

An alternative approach that does not use a tablet device is implemented with the take-picture cardboard tool. The cardboard comes with a printed camera symbol and a barcode that allows its identification. When the take-picture tool is put close to a screen marker on the table, a projection will indicate the selection of that marker and a three-second-countdown animation is started (compare Fig. 4.4). If the cardboard is moved away from the marker, the countdown will be aborted; else, when the countdown comes to its end, the system takes a photo of the surface and the selected screen is updated.

***Process of Defining User Controls and Interface Functionality***

User controls are standard interface components that play a key role in the user interaction. Blended Prototyping supports 7 different types of user controls that are described in the following list.

(1) *Buttons*

Buttons are active areas that are used to react to touch commands by the user, usually to switch to a different interface screen. For this reason, linkage paths can be specified for a button. A button can be defined as a rectangle with free position and size. In the running prototype buttons are usually invisible. This is for the reason to


**Fig. 4.4** Using the photo taker marker

preserve the sketched design of the interface as much as possible to facilitate the advantages of the sketchy look of an interface described above.

Apart from *buttons* and *gesture listeners* all other user controls used in Blended Prototyping are represented in the same design, as in the later high-fidelity Android app design. These controls provide visual feedback to the user; for example a checkbox shows a checkmark when selected, or a textbox may open an onscreen-keyboard to let the user type in text. This visual control feedback could be supplied in a sketchy-looking design.

However, since in the design-processes of Blended Prototyping designers can use whatever tools they like, the pre-defined sketch design can easily diverge with the design of the actual prototype. Such a clash in the sketch design looks awkward and tends to confuse a test user in the interaction. As a consequence, designers will adapt their design to the pre-defined user-controls and are therefore limited in their free expression. To avoid this, Blended Prototyping uses high-fidelity user controls for the components of dynamic behavior [11].

(2) *Gesture Listener*

Mobile devices provide multi-touch screens that allow controlling software not only with simple touch commands, but with a whole range of single and multi touch gestures. Such gestures are an important instrument for a diverse and natural interaction design and should therefore be considered already in user interface prototypes.

Just like buttons, gesture listeners are defined in the prototype as an invisible rectangular shape, free in position and size. Gestures can be assigned to such an area, where upon, just like for the buttons, screens are defined the gesture will lead

to. Available gestures are swipe-, scroll-, and zoom- gestures. The gestures can be either selected from a menu, or demonstrated on the tablet device.

(3) *Textboxes*

Textboxes are used to allow users to type in text, or to present dynamically generated text to the user. Textboxes are defined with their position and size and are presented as a regular native Android textbox in the prototype. When a user touches a textbox on the screen, a keyboard will open up that is used to enter text.

This way, the user interacts with the textbox, just the way she is used to.

(4) *Checkboxes and* (5) *Radio-Buttons*

Checkboxes and radio-buttons are standard user interface components that are used in selection dialogs. In difference to checkboxes, radio-buttons are defined in a group that only allow for one possible selection. Hence, when one member of the radio-button group is selected, all other members of the group are automatically deselected. In user interfaces checkboxes and radio-buttons are usually presented in a fixed size. For the Android platform, Google tested different control sizes in user tests and developed a method that is adjusting the size of the control to the specific hardware conditions. Blended Prototyping sticks to this idea since it does not want to encourage designers to contradict such design rules in their prototype and to eventually end up with a problem in implementing the design in the later software.

Hence, checkboxes and radio-buttons are defined solely by their position. For radio-buttons an additional grouping parameter must be specified.

(6) *Image-Container and* (7) *Video-Container*

Image and video containers are both freely positioned and sized as rectangular objects in the prototypes. They put developers into a position to easily display digital imagery in the prototype. This content can be gathered from a resource that is delivered as a part of the prototype data, or by embedding a link to a resource present in the Internet. In the running prototype, video containers provide an action bar, with the common mechanisms to control the playing of the video.

The definition of user controls can be done with two alternative approaches: first, by using the tabletop control app, and second with the help of low-tech tangible tools. In the tablet-based approach, in the first step, the screen the user wants to define the controls for is selected from the drawer bar on left side of the screen. Now, the latest digital screen version is displayed on the tablet screen. Choosing from the top menu, the user can position and size the user control on the sketch with a simple drag gesture. Checkboxes and radio-buttons can only be defined by their position and not their size.

Long-pressing the controls invokes an onscreen selection dialog, with which the user can remove a control, or select further dialogs, in which the control-specific properties can be defined.

The tablet app is connected via a wireless local network with the tabletop application. It shares all changes in the prototype data with full-duplex enabled


**Fig. 4.5** Palette tool (*left*) and rubber tool (*right*)

web-socket communication protocols. This way, all changes done on the tablet are instantly regarded and displayed in the tabletop application as well, and vice versa.

The low-tech positioning and sizing of controls is done with highlighting pens. In this scenario, different highlighting colors are used to mark the areas in the sketch that should be interpreted as a certain user control.

Highlighters use well distinguishable colors that are usually not used in the regular interface design. The process of the highlighter based recognition of user controls is started by putting a palette tool onto the table, where the colors that should be used in the recognition process are drawn into reference boxes (see below: Fig. 4.5, left). This way, the users of the design tool are free to use whatever colors and pen brands they want to and can select colors that are in enough contrast to the ones used in their interface design. Moreover, with this approach it is important to supply a reference point that is exposed to the same lighting conditions as the color used for referencing the sketches. The highlighting tool was in parts implemented with the help of the student Francesco Bonadiman, who wrote his mater thesis at our chair on a related topic [52].

Users can mark hand-drawn controls in the sketches by coloring the relevant areas with the color they assigned to the corresponding control-type. Just like in the process applied for the photo taker, the ***palette-tool*** is referenced to the closest screen marker. When the users want to start the user-control interpretation process, the screen bearing the highlights is positioned close to the marker tool, where upon an animated countdown is started, on which end the sketch is photographed. Now, the photo is interpreted with a color recognition algorithm that searches the photographed sketch for the colors referenced on the palette card. To determine the user controls in the photographed interface screen, an algorithm calculates the closest rectangles around the found color spots.

For not to staining the physical sketches with bright colors, semi transparent sandwich paper is supplied for the markings with highlighting pens. Beside that, the semi-transparent paper blurs the physical sketch and therefore improves color recognition result of highlighted areas. For the specification of further

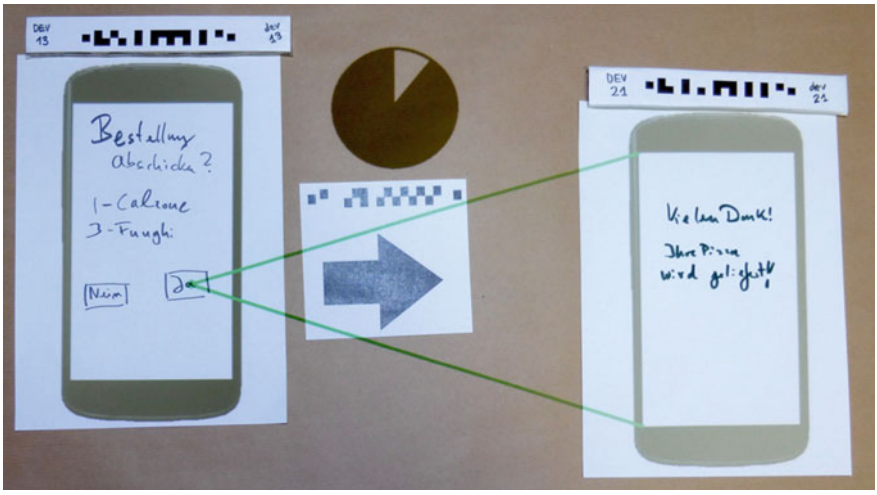


**Fig. 4.6** Connection tool

control-specific properties, other tangible marker tools are used. Connections between buttons and other screens are created with the help of a connector-tool, by putting the tool between the button and the target screen (compare below Fig. 4.6). Groupings for radio-buttons are set by default, when they are all photographed at the same time in a recognition step.

The deletion of user controls is done with the ***rubber-tool*** (compare above Fig. 4.5, right). When a screen is positioned close the rubber tool, a reference to one of its user controls is shown with a red arrow. When the rubber-tool is moved, this reference will change accordingly. Like in the tools described above, a countdown timer determines the deletion of the control. The rubber tool can be applied to the background sketch of the interface screens as well, which makes the system roll back the interface sketch to its previous version.

For creating a connection between a button or gesture-listener and a target screen, the ***connection-tool*** is used. When put on the table, a green dot is projected left of the physical tool onto the table surface. When the cardboard is moved, the dot is moved accordingly. A connection is defined by positioning the dot above a button and ate the same time positioning the target screen at the tip of the arrow printed on the connection-tool (compare Fig. 4.6). Now, a countdown will be started, upon which end the connection is created.

Another low-fidelity control card is the ***copy-tool***, which copies content from one screen to another. It contains two trigger areas: a first one for the source and another and a second one for the target of the copy action. When two screens are identified in the trigger areas, and are left there until a countdown completion, the content of the source screen is transferred to the target screen.

Further definitions, which are necessary for the other advanced controls are cumbersome to achieve with low-tech measures. Links for pictures or videos are

hard to set up without even having a keyboard. And the demonstration-based definition of gestures is difficult to implement with low-tech tools as well. I experimented with a number of methods, like supplying a virtual, marker based keyboard on the table surface, but decided against their integration into the system. After all, I did not want to implement skill games, but a design tool that is applicable in productive work.

Therefore, users must use the tablet devices to define such controls, or define the properties manually in editing the generated prototype code.

### *4.2.3 Module 2—The Creation Tool*

As displayed first in a MobileHCI PID-MAD workshop paper [9], two alternative processes can be used to generate prototypes on the basis of the data previously generated in the design tool: A fully automated process, where a prototype is created without further ado with the speed of a button click, or in a semi automated process, where additional programming code can be added to the prototype in an intermediate step.

With the automated process, prototypes are created as quickly and easily as possible. However, prototypes generated in this fashion are limited in their functions to mere click dummies: prototypes that do not provide more than static interface screen changes, each triggered by a button or gesture listener. This does not mean at all that such prototypes are useless. Click dummies are a widely used approach in early design stages, where the benefits in speed to provide quick testable prototype versions outweigh the ability to regard complex behavioral aspects. This is why many of the approaches discussed above concentrate their efforts on supporting the development of click dummies. In addition to that, designers can use the automated prototype generation feature to try out their prototype ideas, the minute they created it. Just like in the paper-based prototyping approach, ideas can therefore be quickly tested and adjusted in a prototype, without being distracted from the collaborative ideation process for too long.

However, as shown in the results of the expert reviews described in Chap. 3 above, the type of questions that are addressed with a prototype do change in the course of the development process. Though in earlier states prototypes serve the purpose of simple demo cases to provide general feedback, in later states more specific questions on more specific prototype aspects will arise, or the experience in long-term usage tests gets in the focus. For this, a more elaborated functionality becomes necessary.

In the semi-automated process such elaborated functionality can be regarded in contributing additional programming to the code. The code used in Blended Prototyping is written in the programming language Java. This language is native to the Android, meaning that apps programmed for this platform are developed in that same language. Using native programming languages brings two major advantages: First, the same techniques and programming libraries can be used as in the later

product development, wherefore the possibilities and boundaries of the technique are the same as in the later software. Second, the use of native code makes it easier to reuse the efforts invested into the programming code in the product, and hence to follow an evolutionary prototyping approach. To allow for an easy adding of programming code, Blended Prototyping generates classes that are custom designed with a simplified class structure and well marked and commented areas that programmers can use as an entry point to add programming code.

(1) **Using the Automated Process**

In the automated process, no additional functionality is described in the course of the prototype creation process. Therefore, prototypes generated in an automated manner must be defined completely within the design tool. This means that all user controls that trigger a swapping of screens (buttons and gesture listeners) have to be defined before in the design tool.

In the automated prototype generation process this information is used to write the according event-listeners into the programming code of the prototype. The automated prototype creation is triggered automatically, whenever the prototype in the design session is saved. This way, the prototypes can always be tested on the mobile device present on the table without further ado. If a prototype is saved, which code has been edited, these changes will be preserved (compare description below).

(2) **Using the Semi-automated Process with Code Editing**

The app programming code can be edited to describe more advanced features of the prototype behavior. Blended Prototyping offers its users easy processes for the code editing in two different ways: First, to change the code that is run and interpreted in the app on the mobile device itself, and second, for code that can be run on an external Java server.

Mechanisms to easily regard server functionality in the prototype are provided for the reason that most apps do not operate solely on the mobile device, but are connected to servers that provide data or processing services. This way, the powerful resources of server computers can be used for complex algorithms, data security is better established, or big data sources can be accessed.

In the following text, the use of the two different ways to use the semi-automated code-editing functions is explained, first with regard of solely editing app-side programming code, and second, with the involvement of server method calls as well.

(2.1) ***Editing Prototypes Solely on the Client Side***

Prototypes for which code is edited on the client side can regard variables and functions in their processing and connect them to the properties of its user controls.

In the following, the use of the mechanism is described in a simple example dialog, where an email-address is registered for a newsletter.

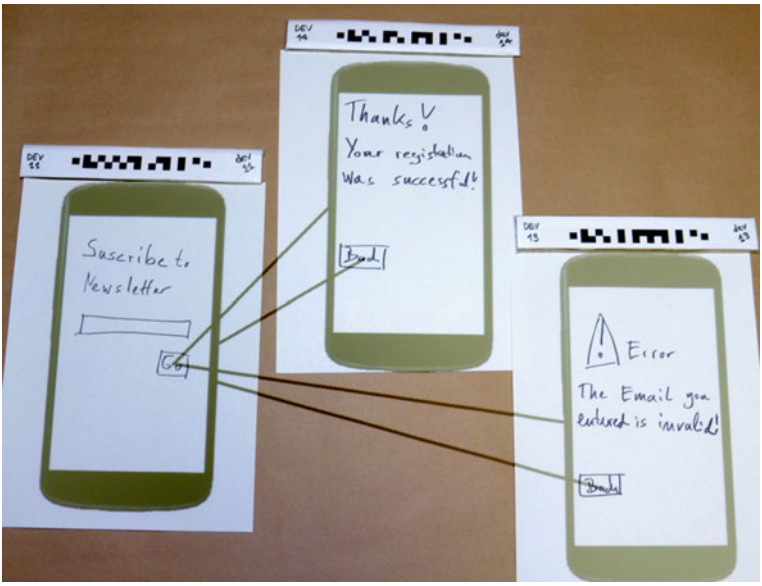


**Fig. 4.7** Prototype example

The example is built out of the three screens that are displayed in Fig. 4.7. On the first screen, the screen to the left, a textbox, a button and a checkbox are defined. The basic idea of the interface is it to offer the user a textbox on the first screen to enter an email address to receive the newsletter, along with a checkbox with which the users indicate their agreement on the newsletter's terms and conditions. Depending on whether or not the checkbox was checked, the user is supposed to land on the upper right screen, which is giving a success message, or respectively on the lower right screen, where the user is informed about the necessity to accept the terms and conditions. For this, the button on the first screen is connected to both other screens, and the button labeled 'back' on both screens to the right are connected back to the first screen.

Whenever the project is saved in the design tool, the automated prototype creation process is started. This means that for each screen that was present on the table, a Java class is generated, where handlers are implemented for the controls the screen contains. Moreover, the classes are stored in an Eclipse project, so that they can be comfortably imported into the Eclipse IDE.

In this integrated development environment (IDE), a package with the name of the prototype will be found that includes a generated class for each of the screens. The class generated for the first screen will contain a local variable for each control that was defined for the screen within the design tool. This way, the user controls can be accessed from the programming code. Moreover, the class for the first screen

in the example will contain a button handler, which is structured in the following way:

```
public void handle_btnScr11_1() {
   if (true) { // TODO: …
      Screen14 targetScreen = new Screen14();
      // Edit code here (start)
      // Edit code here (end)
      // TODO: Define target screen properties here
      delegateToBase.changeToScreen(targetScreen);
   }
   if (false) { // TODO: …
      Screen13 targetScreen = new Screen13();
      // Edit code here (start)
      // Edit code here (end)
      // TODO: Define target screen properties here
      delegateToBase.changeToScreen(targetScreen);
   }
}
```

The code fragment above contains two *if clauses*, one for each of the screens, the button is linked to. Within each of these clauses, the according Screen object is instantiated and a method of the interface-object *delegateToBase* is used with the screen, to cause a change of the screen that is displayed in the prototype. The interface-object is automatically generated in the classes. Please find a full printout of the generated classes for this example attached to this work in Appendix A.

When no changes are made to the conditions of the *if clauses*, the interface will follow the default way of linking to the screen that was defined first as a linking target (in case of the above snippet Screen14).

Changing the interface behavior to regard the validity of the entered Email is easily done with changes highlighted in yellow color in the following fragment:

```
public void handle_btnScr11_1() {
   if (emailIsValid()) { // TODO: ….
      Screen14 targetScreen = new Screen12();
      // Edit code here (start)
      // Edit code here (end)
      // TODO: Define target screen properties here
      delegateToBase.changeToScreen(targetScreen);
   }
   else { // TODO: ….
      Screen13 targetScreen = new Screen13();
      // Edit code here (start)
      // Edit code here (end)
      // TODO: Define target screen properties here
      delegateToBase.changeToScreen(targetScreen);
   }
}
```

For the manual compilation of the prototypes, Blended Prototyping includes a small desktop computer application named *Prototype Creation Manager*. With this software the step of the class creation and the compilation of edited classes to Dalvik VM specific byte code can be controlled manually. The software also informs about errors that occur in the process.

If the design work on the prototype is further progressed and a new prototype version is saved again, a repeated class generation does not simply overwrite the existing classes, but preserves the added code. For this, in each generated class areas are marked with '*//edit code here*' comments, where developers can enter their additional programming. Such areas are present in each generated block: in each method, constructor, member description, handler, and so forth.

When the class generator algorithm finds existing classes for the prototype, it cuts out these comment blocks from each block, and inserts them into the newly created classes. If a block should be not existent in the new version, e.g. because a user control was deleted, the old programming block is commented out and pasted to the end of the new class. This way, valuable programming work cannot be lost accidentally. For the matter of a good programming practice, developers should however not write too much code into the generated classes, but refer to classes they added themselves to the programming package.

### (2.2) *Editing Prototypes on Client and Server Side*

Blended Prototyping includes processes for a simplified embedding of server-side code in the prototype implementation as well. This is done in two simple steps.

First, a *jar* file is created that contains the compiled external server classes. The entry point for the call of the classes needs to follow a public static *Object-to-Object* signature. Being the fundamental class type in Java, *Object* is inherited by all other possible Java classes.

In the second step, the *jar* file is copied to the *ExternalClasses* folder of the Blended Prototyping server and the contained methods are registered in the provided *dynLink.csv* file. Here, the name of the jar *file* is entered along the method name and its package path. For each of the registered methods, the Blended Prototyping test server dynamically generates an path-extension that follows the following pattern:

*createdPath-Extension* = Jar-file-name /package path . class name . method name

These URLs are used to address the code running on the server side from the test app. For this, a method called *callExternalServerFunction* is used, that is provided in the interface *IF_ScreenToAndroidBase*. This interfaces is instantiated in each generated screen class in the class-variable *delegateToBase*. Two parameters are handed to the method: a *String* containing the URL extension generated by the server, and an *Object* that is forwarded as an input to the server method. The method's return type is of *Object*, which represents a direct forwarding of the server method output.

The mechanism implements an *Object-to-Object* function mapping, leaving developers the biggest freedom to exchange whatever object type they like. However, since the objects are transformed to an *ObjectStream* to be sent via an *http* connection, the applied objects need to implement the interface *java.io. Serializable* or *java.io.Exernalizable*.

The following code fragment shows an example of a server-side method call, in which an image is searched for faces. The algorithm behind this search might be very complex and time consuming to calculate on a mobile device, wherefore it is ran as a server function. In the example, a Bitmap is sent to the server function, which is checked where upon an array of points shall be returned, containing the position of recognized faces.

```
package p1.p2;
public class C {
   public Object recognizeFaces(Object inc){
      Bitmap bitM= (Bitmap) inc;
      Point2D[] faces= recognizeFacesByServer (bitM);
      return faces;
   }
}                                                              (Server)
```

Assuming that the class above is compiled to a jar file named *Example.jar*, the client can now use the server method in the following way:

```
...
String url_S =   "Example.jar/p1.p2.C.recognizeFaceByServer";
Bitmap picWithFaces = getPic();
Point2D[] facePositions= (Point2D[])
delegateToBase.callExternalServerMethod(url_S, picWithFaces);
drawFunnyHats(picWithFaces,facePositions);
...
                                                               (Client)
```

### 4.2.4 Module 3—The Testing Tool

The Blended Prototyping platform includes a set of tools that support the design team in the efforts of distributing the prototypes to an audience of testers, to run the test, and to acquire and analyze data about the prototype usage.

For this support, Blended Prototyping provides a *test server*, which first, handles the distribution of prototypes to the test clients, and second, collects the generated usage data.

After the prototype creation process is finished, the prototype data is stored in a folder that is accessible by the server. Now the prototype is registered and allocated to test users. This is done in a simple CSV table, where user credentials can be determined and allocated to certain prototypes.

The prototypes are tested directly on Android devices. For this, a test user first has to download a prototype-testing app, which will later host the native prototype. Within this app, the user enters a username-password combination, where upon the app will connect to the test-server, to check for prototypes that are available to the user. The server checks the CSV table described above, and provides the user with a selection of the prototypes, she is allowed to test. If the user is allocated to solely one prototype, the app starts that prototype directly. Now, the prototype is started, nested within the prototype-testing app as a native Android application. This means, that the compiled prototype code is executed, without actually being a part of the initially downloaded prototype-player app.

The photograph in Fig. 4.8 shows an example of a prototype running on a mobile device. The example shows a screen of a navigation app prototype. The code-snippet on left side of the figure, shows the code edited for the previous screen, where a destination address for the navigation that was entered in a textbox is used to generate a bitmap, which is the loaded in the succeeding Screen14. On this Screen14 the user is provided with a map section that suits her entered destination address. The widgets are of classes starting with the marker 'W_'. This is used in the Blended Prototyping class structure to mark widgets used in the platform, which themselves inherit from native Android-controls. An in-depth documentation of the class-structure is found in a Javadoc delivered with the platform programming code.

Whenever the user starts the prototype player app again, the newest version of the prototype addressed to the user will be downloaded and run without further ado. This is handy for the test-user, since she does not have to reinstall the app at any point; and this is handy for the developers, since they can alter the prototype, swap it with a new version, or allocate the test-user to a different one without the users' notice.

If the user touches the screen while the prototype is tested, this interaction is logged by the app and stored in an internal SQLite database. After the test, these usage data logs are aggregated and sent to the test server. Here the data is stored in a CSV file, which allows for an easy import in software solutions to conduct further statistical analysis. For example, such data could be processed into a video, where the user's interaction with the interface is displayed. Or more specific questions could be answered, such as what average time users stayed on certain screens.

All system components are written in Java, but are compiled for different Virtual Machines. The server of Blended Prototyping is run in the standard Java Virtual Machine (JVM), which is available throughout all main system platforms. As explained above, the prototype data that is interpreted by Blended Prototyping mobile player, is also written in Java.

```
public class Screen11 extends TTSMIPPS_Screen {
    final static String imagePath = "Screen11.png";
    final static String prototypeName = "HelloFusion";
    IF_ScreenToAndroidBase delegateToBase;
    W_Button scBg_11_btn_1;
    W_TextBox scBg_11_tb_1;

    public void handle_scBg_11_btn_2() {

        if (true // connected to one singular screen, so set to true
        ) {
            Screen14 targetScreen = (Screen14) delegateToBase
                    .getExistingScreenObject(new Screen14(getContext(),
                            delegateToBase));

            String address = scBg_11_tb_1.getText().toString();
            Bitmap bm = protoUtil.getMapsImageForAddress(address);
            targetScreen.scBg_14_pic_1.setImageBitmap(bm);

            delegateToBase.changeToScreen(targetScreen);
        }
```

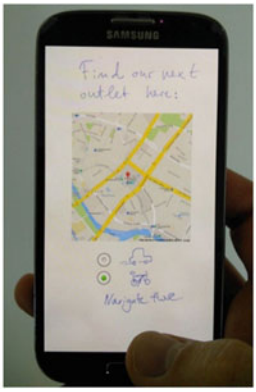


**Fig. 4.8** Example of a running prototype (*left* code from the previous screen, *right* photo of the prototype running on the device)

The interaction between the Blended Prototyping components for the prototype generation works by storing the necessary prototype data in a folder accessible by the server and in editing the CSV table. Therefore it is not necessary to run both applications on the same machine or at the same time.

The communication and data exchange between the server and client app is however established via the Internet using HTTPS.

## 4.3 Design Decisions in the System Implementation

In this section, selected design decisions in the implementation of the Blended Prototyping system are explained. The description is separated for the system models. First design decisions in the design tool are explained, followed by that, aspects of the creation tool, and finally specifics of the testing tool are highlighted.

### *4.3.1 Implementation of the Design Tool*

#### (1) Hardware Setup of the Tabletop-Computing Environment

The tabletop-computing environment used for the Blended Prototyping design tool uses the following hardware components: an EPSON THW-5200 video projector, a Canon E650 DSLR photo camera, a Logitech HD Pro C920 Webcam, different Android tablet devices (Nexus 7 and Samsung Note 10.1), a regular desktop computer running the tabletop control software, and a Cisco LinkSys E4200 WLAN Router to establish a high speed connection with the mobile devices used in the tabletop context. Beside these electronic components the design tool

uses a big tripod that is able to host projector and cameras above the table surface, a regular table as a working space, and paper and pens that are used to create the user interface sketches.

All of the used hardware components are mid-performance off-the-shelf solutions that are available for very reasonable prices. The employed video projector supplies full-HD resolution at a good brightness level (1800 ANSI), its market price as for the end of 2015 is about 1000 €. The Canon DSLR is a standard-consumer camera with an image resolution of 18 megapixels. The standard 18–55 mm f/3.6-5.6 zoom lenses are used that came in the camera kit package. Currently, the camera price ranges at about 350 €. The employed desktop computer is a standard machine with no distinguished performance capabilities in any respect. The employed webcam is a full-HD standard webcam for about 70 € and the other hardware used does not need to reach any significant performance either. In sum of approximately 2000 € should be sufficient to reproduce the setup.

Both cameras and the video projector are assembled on a tripod in a way that their lenses are targeted at the table surface. That way, the video projector supplies an output channel, the cameras an input channel to the user. The Android tablets are connected to the computer system via a shared WLAN and can be used as one of different alternatives to control the application. The provided smartphone can be used to test prototypes created in the test sessions immediately.

The supplied paper sheets come with a printed marker, which can be recognized by the system in the images produced by the system's cameras. The pens used in the sketching process should create a clear stroke, which does not necessarily needs to be very thick. The highlighting pens, used for the marking of user controls, should have strong colors that are well distinguishable from another.

(2) **Software Implementation of the Design Tool**

The Blended Prototyping design tool was entirely programmed in Java. Though Java does have its downsides when it comes to high performance algorithms, two major advantages of the technology determined my decision. First, programs written in Java are compatible to an all major operating systems. Java programs are run in a virtual machine, a sandbox that is installed on the operating system to provide the Java program with the same conditions in each environment. Therefore, Blended Prototyping can be principally run on every operating system, be it a Linux distribution, Windows, or MacOS.

As described in more detail below, for yielding higher performance results, some third party software packages are used for the camera control and real time image processing. All these packages are compatible to the major operating systems named above.

The second and maybe even more important reason for me to use Java as the core programming language of the Blended Prototyping platform is its popularity. Java is a standard object oriented programming language and is taught regularly in schools and colleges. It is obviously very important to employ technologies in the development of a system that allow others to easily contribute to the work. This

way it was comparatively easy for me to introduce student workers into the programming code, when I asked them to contribute some additional features.

Beside that, I plan to make the platform available open source. With this I do not just hope to provide my developments to interested users, but to find developers that get involved in expanding the platform and in adding new modules. This is far more likely to happen when the used technologies are popular and easy to use.

The system is currently run on an off-the-shelf standard stationary computer with no special specifications. Probably, the system would run with a more efficient use of calculation resources if it had been programmed with a technology closer to the hardware, like C++. However in Blended Prototyping, highly performance intensive calculations, like the ones that occur in image processing tasks, are done with the help of the javaCV framework. This technology accesses hardware resources like the graphic cards GPU to yield higher performance. Therefore, resource issues have never been a severe topic in the use of Blended Prototyping and the system runs stable without perceivable lack.

In the programming of the design tool the Model-View-Controller (MVC) design pattern [118] was applied. The concept argues for a strict separation of concerns in the programming, by providing separate classes for the data handled in the software (models), separate classes that provide a view to that data (views), and separate classes that are used to manipulate and manage the data (controller).

The implementation of this design pattern provided a better overview of the created class structure and made it easier to build different views to the same data basis. The models were moreover implemented in a way that allowed them to be directly serialized into JSON objects and byte-streams.

In the design tool, the projected screen content is bound to the physical paper counterpart: When a paper sheet is moved, the projected content follows the paper screen in its exact position and rotation. To implement an according graphical representation object, a solution had to be found that allows the organization of different content in freely positioned and rotated graphical containers.

The implementation of such graphical containers went through a long history of different approaches, ranging from attempts to adapt principles of Java Swing, to Java AWT, where upon an open source third party framework called Piccolo2D was used. That framework however had a couple of bugs in itself and was not continued to be supported since 2013. Luckily the framework JavaFX developed vividly in the past 5 years into a well documented, high performing, and stable version that is since Java 8 even part of the standard Java development kit. Therefore, the whole graphical backbone of the Blended Prototyping design tool had been shifted to JavaFX. This technology does not only provide a better performance and easier implementations of dynamic user interface content. Today JavaFX is moreover a well-known and used technology that will be supported within the Java Framework in the foreseeable future.

The communication between all the devices used in the design tool is done via a wireless network, emitted by a router that is part of the hardware setup. All relevant prototype data is shared in serialized versions of the regarding screen models in JSON structures. The exchange of the objects is conducted within a communication through web-sockets. This protocol allows for a full-duplex communication and steadily refreshed prototyping data on all devices.

The serialization of the models to JSON structures was implemented with the help of Jackson parsers.[1] The implementation of the web-socket communication was done with the framework Autobahn-WS.[2]

(3) **Choice for Physical-Objects AND Tablets**

The Blended Prototyping design tool follows two alternative interaction approaches to command and control the tabletop-computing environment. These are on the one hand the table control app that is installed on tablet devices, and on the other hand low-tech tools, based on paper and other physical content. The reason for the decision to follow both approaches is, that either of them has its advantages and disadvantages.

The tablet apps play out their strengths in reliability, precision, and flexibility in the supported input methods. The sketches opened in the app can be fluently zoomed so that the positioning and sizing of controls can be done very precisely. The techniques used to achieve this are well implemented in the Android platform and well used in many other mobile apps. Moreover, most users will be well used to the typical interaction commands in mobile applications. Therefore, misinterpretations of the users' input commands happen rarely. In addition to that, the flexibility of the input methods on mobile devices is immense: Keyboards to enter text can be used effortlessly, the definition of finger gestures to connect to the prototypes gesture-listener controls can be demonstrated easily, and the device cameras can be used to create easy references to the screens on the table.

However, the big downside of the tablets is that they risk disturbing the collaborative work at the table. Tablets are not well suited to be used in groups, since only one user controls them at a time. This user tends to narrowly focus on the interaction with the app, and stops being part of the collaborative discussion.

In comparison to that, low-tech techniques that are based on paper and cardboard are more easily used as shared tangible objects in the collaborative group interaction. Users applying such techniques do it in a more open way, allowing others to participate more easily. However, the input is far more limited when it comes to precision or flexibility of input methods. With low-tech tools, no keyboard can be reasonably implemented. Gestures to describe the user control of a gesture-listener cannot be simply demonstrated, but has to be chosen from lengthy lists.

Overall it has to be regarded, that Blended Prototyping is not a toy, but a tool for professionals working on a task with a lot of time pressure. This being said, it has to

[1] https://github.com/FasterXML/jackson (last accessed 11th April 2016).

[2] http://autobahn.ws/ (last accessed 11th April 2016).

be emphasized that all the tools in the process have to work smoothly, without interrupting the designers from the group discussions and from formulating their creative thoughts.

A feasible alternative that combines both advantages of the discussed techniques could be to use large multi-touch tables as a basis for the design tool. However, such devices are costly and to implement stable interaction techniques for a multi-user use is far from trivial. Besides, the blending of physical and overhead projected content could not be transformed to multi-touch screen applications.

(4) **Marker Tracking and Image processing**

***Barcode Markers as a Reference System***

In the Blended Prototyping design tools barcode markers are used to identify paper sheets and to determine their exact position and rotation on the table surface. The employed barcodes are initially based on previous work by Rohs and Kratz [84], who used the technique for linking personal mobile devices to tabletop surfaces.

The employed cameras are connected and controlled to the design tool with techniques, which were selected with respect to their compatibility to other hardware. The webcam is read with the javaCV CvCapture that supports any standard webcam. The Eos-DSLR is controlled with the use of the open source project gPhoto2,[3] which is a free digital camera control software that currently supports more than 2100 camera models.

Each of the system's cameras, as well as the video projector, has its own perspective at the surface and applies an individual coordinate-reference system to process the images. These coordinate-reference systems are created within the system calibration process.

The calibration is done with the help of barcode markers that are projected at different locations on the table surface (compare Figs. 4.2 and 4.3 on page 80). The employed cameras record these markers, where after their presence and position is recognized in the delivered pictures. Now the position of the markers in all three coordinate systems is known, a warping system is created that is able to transfer one position on the table from one device coordinate system to the other (compare [84]).

***Coping with Camera Distortion Effects***

The cameras employed in the system deliver images that are negatively affected by two optical effects: perspective and lens distortion. These distortions have to be dealt with in both, the marker tracking and the digitization process.

Perspective distortion effects are a result from taking images of an object from a different then perpendicular perspective. In such images, not all objects of the layer

[3]see: gphoto.sourceforge.net (last accessed 11th April 2016).

have the same distance to the camera lens, so that closer objects are pictured bigger than such farer away.

In the demonstrative Fig. 4.9, two cameras are aimed at a rectangular object on a table surface. The camera $C\text{-}_{ideal}$ is attached perfectly orthogonally above the table, whereas the camera $C\text{-}_{shifted}$ is tilted away from the. The image results for the two cameras are displayed in a schematic way on the right side of the figure above. Where the image of the ideally positioned camera would display the red object as it is, as a perfect square, the image of the shifted camera would show the red object as a trapezoid.

In the actual hardware setup of the Blended Prototyping design tool, the shifting angle is not as extreme as displayed above. However, in practice, the shifting is not limited to one axis of the Cartesian room coordinates as in the example. Therefore a perfect square is usually recorded in form of a random quadrangle.

Perspective distortion can be corrected in different ways. If the exact positioning of the camera in relation to the table surface as well as the exact setup of the camera lens is known, the perspective distortion could be equalized in geometric transformations. This solution however limits the flexibility of the hardware setup to an extent that is hardly feasible in the context of the Blended Prototyping tool. Even small changes in the positioning of the setup will have a severe effect on the quality of the calculations.

A second way to correct the perspective distortion can be established with a polygonal shape warping of the photographed objects. If a reference object is put into a photo that bears perspective distortion, algorithms can be used to stretch and

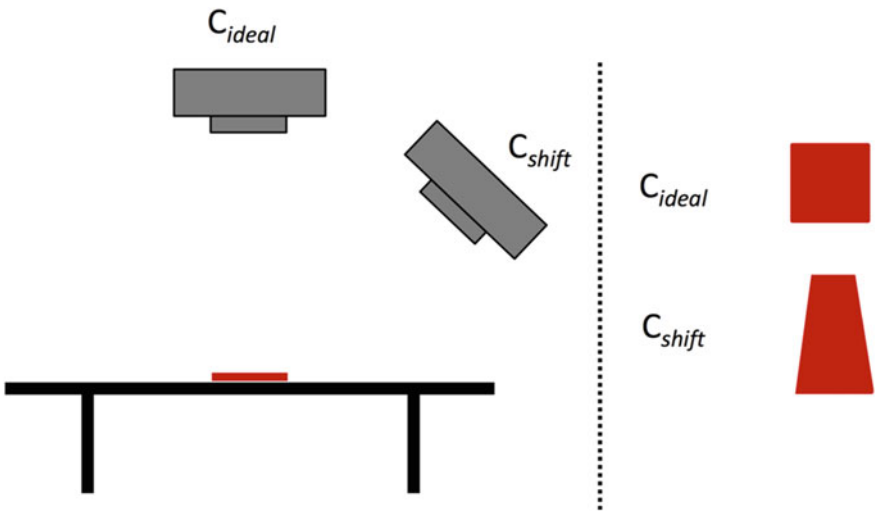


**Fig. 4.9** Perspective distortion—examples for camera positioning (*left*); schematic distorted pictures (*right*)

crush the image in a way, that the expected shape of the reference gets restored. The barcode markers used in the Blended Prototyping design tool can serve the purpose of such a reference object.

Another source of distortion in the images arises from the imperfect optical construction of camera lenses. Especially lenses of short focal distance often cope with barrel distortion effects, where actually straight lines at the image border are recorded bended outwards on the photo.

Correcting lens distortion is well discussed in the scientific debate and achieves good results. However, the applied mathematical methods are far more complex, than the ones discussed above for the correction of perspective distortion effects.

Polynomial based methods are available that aim at estimating the distortion parameters can be found in the work of Jung et al. [61]. Ahmed and Fang [3] propose a non-metric method of calibration that finds distortion parameters and refines them using non-linear optimization; yet others [16, 42] follow non-polynomial methods founding on logarithmic distortion that are based on the way, wide angular lenses are constructed. However, each approach requires on sacrifice: either in accuracy or in processing time. It should be carefully considered, which of the factors is more important for the specific application scenario [152].

In the Blended Prototyping design tool two image correction scenarios have to be considered. First, the interpretation of the video signal provided by the webcam to track the paper sheets present on the table, and second, the transformation of the photos taken by the DSLR camera for the digitization of the interface sketches.

The tracking of paper sheets runs steadily as a parallel process. If the image processing time in this context is too slow, the movement of the paper sheets on the tabletop will not be recognized fast enough to let their connected projection follow the paper sheets seamlessly. This affects the usage quality heavily; hence, in this context the fast performance of the processing should be the main motive.

The mixture of distortion effects described above makes their mathematical correction in the webcam image complex and time extensive. Testing the implementation of the approaches discussed above slowed the tracking down to an unbearable extend.

For this reason a rather simple, but very time effective solution was implemented in Blended Prototyping, which helps to handle both distortion issues at the same time.

The approach uses a second calibration step after the global calibration described above, where twelve smaller local calibration systems are projected as a mesh on the table. For each of these local systems its own warping system is created using the same technique as for the calculation of the global warper. Since in the local systems the local distortion effects can be regarded in higher detail than in the global system, the results of identifying a marker position in a local system is much improved (Fig. 4.10).

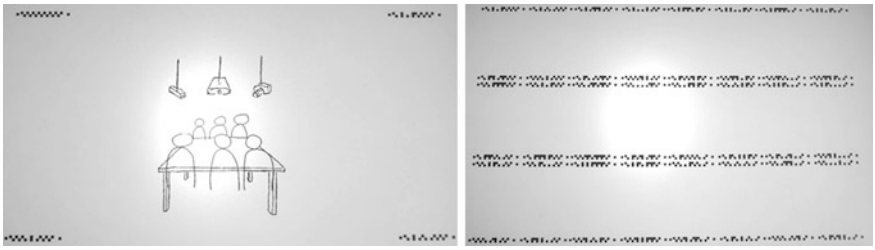

**Fig. 4.10** Global (*left*) and local (*right*) calibration systems

Following this approach, two steps are necessary in determining the exact position of a marker. First, the approximate position in the global reference system is calculated. On the basis of this coordinate, the local reference system the marker is positioned in can be resolved. The application of the according local warping system to warp the identified marker position will deliver accurate results.

In case of the pictures warped for the digitization process the situation is similar but for the fact, that not one but four position points, one for each corner of the sketching area, are calculated. In the photo excerption and correction of the sketched user interface screens, Blended Prototyping uses the marker dimensions as a reference to determine the four corner points of the arbitrary quadruple that shows the sketched image in the photo. Now, a graphical transformation method is used to stretch and crush the quadruple into a perfect rectangular image of the hand sketch.

This proceeding requires a longer processing time, than the simple position and rotation tracking done for the video signal. However, the digitization process does not run continuously, but is triggered rather from time to time by the system users. Waiting a second for the digitization result is therefore less disruptive to the design process, especially since a paper sheet is swapped with a blank one after the digitization step anyway.

### *4.3.2 Implementation of the Creation Tool*

The Blended Prototyping creation tool generates prototypes on the basis on data provided by the design tool. This contains an image file for each of the screens used in the design tool. Beside that, a JSON file is included that explains all aspects that were defined in addition to the sketches.

JSON provides standards for structuring information in nested nodes that is well readable for both, computer algorithms as well as a human observer. The format is a standard file type used frequently in all modern programming languages. Therefore, the technology provides an open interface that others who might want to implement different modules for the prototyping platform can easily connect to. As explained above, the Blended Prototyping design tool follows the Model-View-Controller concept to facilitate a consequent separation of object data, the views on this data,

and the management of its manipulation. This concept makes it very easy to export all relevant object data with a simple serialization with a JSON parser. In the prototype creation process, Blended Prototyping follows a native programming approach.

The generated classes are structured in a way that allows even programmers with little experience to understand the meaning of its components, as well as the regions, where further code editing should be done. For this, the class solely contains instances of customized classes that are inheriting from their complex Android counterpart and create a simplification layer for the programmer.

All available data for a user control is considered in its instantiation automatically. Therefore, all controls are positioned and sized in the correct way, radio-buttons come in their correct grouping, and marked out handlers are created for each button.

The generated screen classes inherit from their own customized class as well. This class establishes all the necessary Android technology to allow the swapping of the screens inside the app. The swapping itself is done with the simple method call *changeToScreen (Screen..)*, which expects any form of a Screen object to swap the interface to. If connections for buttons or gesture listeners were already defined in the design tool, such Screen objects are automatically generated. Now, the programmer can simply address the user controls that are present on the screen with simple getter and setter calls of the screen's member variables.

The experience made with novice programmers using the class structure to implement their prototype ideas was promising. This is related to the workshops that were conducted with school kids, to the programming efforts that were undermined by the Telekom designer, as well as the design student that employed the technique in developing a pedillac booking app.

### *4.3.3 Implementation of the Testing Tool*

Generally speaking, all Android applications are written in Java. However, Android apps have to take into account a number of specific factors that derive from the mobile devices they are ran on. Therefore, Android includes a number of technologies that do not occur in normal Java applications but are specific to the Android environment. Consequently, Android applications are not ran in the standard Java Virtual Machine (JVM), but are interpreted in an Android specific Virtual Machine, called Dalvik VM. Binaries compiled for the Dalvik VM are stored in the Android specific DEX-jar format.

The exchange between the Blended Prototyping components that are involved in the prototype testing is established in two different ways. Where files between the prototype generation and the test server are shared via the file system, the exchange between the test server and the testing app is done via the Internet, using secured HTTPS connections. Using this well-established protocol opens the system architecture to further developments and adaptions.

Controlled by two CSV setup files, the test server carries out all necessary tasks for the evaluation session management, prototype distribution, and user interaction logging storage. In addition to that it offers standardized ways to invoke server side prototype functionality.

The planning of user studies is done in simple CSV tables, where user credentials can be entered and allocated to the prototypes the user is meant to test. I decided to use CSV files for this task and not to implement a similar system in a database, because this way, the whole study management is aggregated in a single file that can be shared and distributed by the development team members, for example by sending it by Email, or copying it onto an USB stick.

Participants of studies download and install the test client on their Android device, which will then download prototype binaries from the test server. The test app captures all the users' input commands, and the controls the user interacted with, along with a timestamp. This data is continuously stored in a SQLite database, held locally by the app on the mobile device.

When the network conditions of the mobile device are stable, this data is uploaded to the test server for later analysis. A continuous upload of interaction commands to the server risks the slowdown of the user interaction. User interaction logging data is stored on the server in serialized SQLite files, as well as in CSV files that allow for an easier further processing of the data by the users.

## 4.4 Discussion of the System Implementation

The Blended Prototyping approach is developed as an answer to the third research question, postulated in Sect. 2.5 above:

RQ 3: *Is it possible to create a new prototyping approach for mobile UIs, which adapts the full advantages of the paper-based prototyping approach, and at the same time adapts advantages of high-fidelity prototyping approaches to produce mixed-fidelity prototypes? Such a system should meet the following motives:*

- *create a platform for interdisciplinary teamwork*
- *provide an approach that supports creative work*
- *provide an approach that is easy to learn*
- *facilitate the advantages of the physical use of pen and paper*
- *facilitate the advantages of the abstraction that are inherent to paper sketches*
- *deliver quick prototype results*
- *blend the paradigms of throwaway and evolutionary prototyping, and allow reusable programming of extended prototype functionality*
- *support on-device tests to allow even large-scale user tests in the real-use context that take into account device related usability problems.*

The concept of Blended Prototyping was developed and transferred into a working prototyping platform, which can be used to create mixed-fidelity prototypes in a new process. The approach aims to adapt the advantages that are frequently ascribed to the method paper-based prototyping, and at that same time allows to create prototypes that are run directly on the target device. For this it facilitates a new design process, which uses a tabletop computing system to develop prototypes on the basis of paper-sketches. Within this design process functionality can be defined for the prototype design, which manifests itself in the definition of user controls. Moreover, the approach supplies simple-to-use processes that allow for additional programming of user interface behavior. Since this programming is done in the programming language that is native to the Android platform, prototype code created in this fashion can be re-used for further prototype advancements and in the later programming of the final product itself. This way, the approach blends the advantages of a throwaway prototyping approach on the basis of paper with the strength of an evolutionary prototyping approach, that uses the same tools for the prototyping, as for the later product development.

The Blended Prototyping design process is based on a tabletop-computing environment that is targeted to *create a platform for interdisciplinary teamwork*. It is a common habit to have collaborative discussions at a table, where a group can sit together, look each other in the eye and directly share information. In a way, tables are the natural habitat of collocated collaborative work. For this reason, as described above in Sect. 2.4, many group-work oriented technical applications have been developed, that use tabletop-computing systems and interactive surfaces as the center of their human-human-machine interaction. In contrast to this, in the collaborative use of desktop computer systems, only one of the users has control over keyboard and mouse, and holds the power to control the application. As Cook and Bailey [36] point out, this gatekeeper effect oftentimes results in a lack of motivation at the other team members, who easily feel isolated from the design process.

However, even in computer systems that use technologies like tabletop-computing systems, an inherent risk exists to disturb the users' discussion and exchange with computer interaction tasks that demand too much concentration. For this reason, Blended Prototyping aims to provide a natural use context that is hosted on a regular table, where the design is focused on regular pen and paper. Designers are free to create and discuss their sketches with close to no system interaction, regarding the system more as a normal table with a bright lamp above it. Each system interaction step, like the photographing of the sketches, or the definition of controls, is designed under the premise to be quick to use and not to distract the user too much from the actual group design process. Moreover, only interaction processes were included in the design tool, that had a very low error rate. A system that misunderstands the commands is annoying and will surely interrupt productive processes. Therefore, high-tech ideas that might be *cool* but are hard to implement in a solid way, were always discarded. For example, this concerns experiments with techniques that establish multi-touch input on the table surface with Microsoft Kinect spatial cameras.

Distracting the design teams as little as possible in their doing is an important criterion in the attempt of Blended Prototyping to *provide an approach that*

*supports creative work*. As pointed out above in Sect. 2.4.2, creativity is the result of complex, high-level cognitive processes that are investigated by a number of research domains' [133]. It is much harder to tell what factors facilitate creativity, than to make clear what factors are likely to hinder the development of creativity.

As Jim Blinn [24] points out, for creativity to happen it is important to have the ability to create a certain degree of chaos. Such chaos is the basis for sorting out ideas that are promising to deliver valuable results. It is important for a tool to allow creative chaos by giving its users the freedom to do and create whatever comes to mind. For this freedom of design, the physical use of paper and pen is an excellent basis, users are usually well accustomed to. In contrast to most computer software, paper will not distract users with error messages, or annoying dialogs that ask for property specifications. Moreover, paper is well suited to meet the four requirements Ben Shneiderman is addressing towards tools that are meant to support creativity: to *support exploratory search*, *enable collaboration*, *provide rich history-keeping*, and allow to *design with low thresholds, high ceilings, and wide walls* [137].

The design-tool of the Blended Prototyping platform operates an overhead projected tabletop-computing environment, where regular paper-sheets are used to create prototypes on the basis of hand-sketches. Therefore, the approach *facilitates the advantages of the physical use of pen and paper*. When physical sketches are converted to a digital version, their content is projected onto paper-screens, where additional drawings can be added to the sketches. This preserves the advantages of physical sketching, even after the first digitalization steps. Therefore, the designers can still benefit from the quickness of hand sketching as well as its improved ability to communicate ideas to other team members.

As pointed out in Sect. 2.3.2, using hand sketches on paper as the central instrument of design, the approach aims to *facilitate the advantages of the abstraction that are inherent to paper sketches*. This abstraction has its primary strengths in communicating the idea to other team-members, as well as asking the most relevant questions in the prototype testing. To adapt these advantages, the design process of Blended Prototyping is focused on physical hand sketches on paper.

However, in the running prototype, some elements of the user interface are substituted with high-fidelity controls. As described above in Sect. 4.2.4 this is the case for those controls that need to provide users with a visual feedback, like checkboxes or text fields. The decision not to use prepared sketch-like looking user interface elements was motivated by the risk, that such predesigned sketch-like design is likely not to match the sketching style of the rest of the prototype and can therefore confuse overall prototype design. The situation is different for those user controls that do not need to provide instant visual feedback, like buttons or gesture listeners. Here, to preserve the abstract sketch design, invisible active areas are used rather then high-fidelity presentations of the controls.

The Blended Prototyping is designed to *provide an approach that is easy to learn*. In its design-tool the approach focuses on a paper sketched design of user interfaces, which in itself is very easy to learn, as displayed in detail in portraying

the paper-based prototyping approach above in Sect. 2.3.2. Controlling the system with the tablet device app should not require extensive reading. The app uses interaction techniques that are widely applied in other mobile apps, like an action bar to trigger commands, a side-side drawer to switch between different screens, or touch and drag gestures to position and size controls on the screens. However, the ordinary tool user will likely have no profound experience in interacting with physical objects to control a tabletop-computing environment. Therefore, for the use of the low-fidelity methods, the user will need a short introduction. To give the users a better orientation using the low-fidelity tools, visual feedback with figures or texts is provided to the users.

To create prototypes in Blended Prototyping, two alternative processes are available.

The automatic prototype creation is done without further action of the user. However, the manual prototype creation process requires the programming of additional code. Here, specific classes were created to simplify the code editing as much as possible. Basic Java knowledge should be sufficient to implement basic prototype behavior.

In the module for the prototype distribution and testing different measures have been implemented, to simplify the process: User-studies and user access rights can be handled simply by editing a CSV file and the usage data generated from tests is automatically stored at the test server in a format that is easy to process in the further statistical analysis.

The requirement to *deliver quick prototype results* is regarded in different aspects of the system design. First, as discussed above in the analysis, as well as in Sect. 2.4.3, focusing the design process on physical paper sketches yields fast design results. Second, the prototype creation process is designed under the premise to deliver fast results. Apart from supplying a process for the automatic creation of prototypes, the manual editing of code is simplified to an extent where it should be able to deliver fast results as well. Finally, as described in the previous paragraph, testing the prototype requires close to no time investment.

As displayed in detail above in Sect. 4.2.3, the developed approach *blends the paradigms of throwaway and evolutionary prototyping, and allows reusable programming of extended prototype functionality*. When editing code for the prototype, some techniques that are specific to the Android platform are ignored. This makes the code editing easier, however, when programming the final product, such technologies need to be regarded. Apart from that, since the programming is done in the language native to Android, the programmed code can be reused to a large extent. Therefore, Blended Prototyping can be used in an evolutionary prototyping manner. However, at a certain point in time of development process, a switch to the standard development tools that allow a truly flawless evolutionary prototyping should be made.

Regarding the last requirement listed in the design objectives, Blended Prototyping *allows on-device tests even with large-scale user tests in the real use context*. As described above in Sect. 4.2.4, the prototypes generated with the approach are tested directly on the mobile device and a technical infrastructure is

deployed that includes processes for the user-management, prototype distribution, and logging of usage data. Therefore tests with larger numbers of users can be executed without a considerable effort.

Prototypes developed with this approach can uncover device related usability problems with respect to most of the external effects pointed out in Sect. 2.4.1. However, since Blended Prototyping concentrates on creating mixed-fidelity prototypes based on paper sketches, limitations lay in the production of user interfaces with particularly dynamic content.

The judgment of many points in the analysis above is made on the basis of comparisons to other approaches and estimations. To verify these considerations, the system needs to be evaluated. A first feedback to the system concept was generated in an expert rating, described above in Sect. 3.2.3. An evaluation of the system performance is done in a comparative user test, which is described in the following chapter.

# Chapter 5
# Comparative Evaluation of Blended Prototyping

This chapter displays the comparative evaluation of the Blended Prototyping approach with two other tools. For this, first the choice of the compared reference tools is explained. Then, in the second section of this chapter, the most important performance indices for the comparative study are identified from the catalog created in Chap. 3, and assessment methods for these indices are discussed. Followed by that, in the third part of the chapter, the execution of the evaluation and its results are displayed. In the final and fourth part of the chapter, the results are further investigated in a short discussion.

## 5.1 Choice of Comparative Prototyping Tools for the Evaluation

The performance of a prototyping tool can only be tested and judged in comparison to other tools. In the choice of suitable reference tools, it should be taken into account that the selected tools are developed for a similar purpose and application context. The tools should therefore be targeted to be used in similar development stages and have comparable design objectives.

As displayed above in Chap. 4, Blended Prototyping was developed under two general design objectives: (1) to put developers into a position, where they could yield testable prototypes as early, easily, and fast as possible, and (2) at the same time be able to form the prototype as complex as needed to answer the exact questions they have in mind.

Therefore, to evaluate Blended Prototyping, two reference tools were selected that each specifically addresses one of these two requirements. As described above in Sect. 2.1.2, the method of PBP is praised for its advantage, to deliver fast prototypes that can be tested with users without much effort. Further, the ease of applying the method is often underlined. It focuses the design process to the use of



B. Bähr, *Prototyping of User Interfaces for Mobile Applications*,
T-Labs Series in Telecommunication Services,
DOI 10.1007/978-3-319-53210-3_5

hand sketches, a technique that is applicable for pretty much everyone without further training. This line of argument is supported by experience made with the PBP method for many years [140]. Therefore I chose PBP as a reference tool for the first general design objective of the Blended Prototyping approach.

As for the second goal of the approach, to supply a tool that allows enough complexity in the prototype to be applicable in later design stages as well, standard IDEs might seem as appropriate reference tools. Such tools are usually used in the programming of the later product; consequently, functions of all complexity levels could be regarded with such tools in the prototype. However, the use of such tools requires profound programming skills and is too cumbersome in early design phases, where prototypes of reduced complexity should be produced in a fast manner. As portrait above in Sect. 4.1.2 the tools currently available that come closest to the possibilities of the mighty standard IDEs, but at the same time focus on a fast and easy creation of prototypes, are tools like Axure, Mockflow, or justInMind. Therefore, to select a reference prototyping tool that fulfills the second overall design goal of Blended Prototyping I decided to use one of these prototyping software tools. In the end, the choice fell to Axure for being a widely used commercial prototyping and mockup software, which bases its prototype design on the layout of different single screens that can be connected to one another with buttons and other controls. Small program snippets can be produced in Axure, either manually of with the help of dialog builders, with which the complexity of the prototypes' function is progressed. This way, the tool allows for more complex prototypes, but at the same time has a low entrance barrier to be learned and applied even by novice users. Moreover, unlike Modckflow and justInMind, Axure grants free software licenses to university research and teaching projects.

## 5.2 Identifying Performance Indices for the Comparative Evaluation

### *5.2.1 Identifying Candidates from the Requirements Catalog*

Above in Chap. 3, a catalog has been developed that ranks the most important requirements towards mobile app prototyping tools with respect to different development stages. The catalog points out a set of 16 requirements that were evaluated with experts, and in addition to that includes 5 supplement requirements that were suggested by the experts. This catalog can now be used as a basis for the evaluation of the Blended Prototyping tool, in comparison to the two other approaches identified in the previous section.

Blended Prototyping provides mechanisms to develop testable prototype designs as early as possible. At the same time, it offers techniques like the inclusion of reusable programming code that support the development of more complex

**Table 5.1** Requirements with mean relevance for very early to middle phases

| Requirement | Mean (v.e − middle) |
|---|---|
| Getting quick prototypes | 4.3 |
| Collocated group work | 4.2 |
| Freedom of creativity and design | 4.1 |
| Simultaneous tests of different ideas | 3.9 |
| Support of expert reviews | 3.6 |
| Support of design reviews | 3.4 |
| Tests in the real use contexts | 3.3 |
| Reusable programming code | 3.3 |
| Remote group work | 3.1 |
| Reusable prototypes | 3.1 |
| Easy setup and distribution of user-test | 3.1 |
| Independent parallel development of designs | 2.9 |
| Use of animations | 2.8 |
| Advanced functionality of a prototype | 2.7 |
| Tests on different platforms | 2.5 |
| Tests with a large number of test users | 2.3 |
| Fun factor (n = 1) | 5.0 |
| Usability of the tools themselves (n = 2) | 5.0 |
| Compatibility of the tool with different platforms (n = 1) | N/A |
| Open source availability (n = 1) | N/A |
| Tutorials and help for the tool (n = 1) | N/A |

prototypes as well. This way, the approach primarily addresses early to middle design stages of app development projects. The search for the most relevant performance indices of the comparative evaluation should therefore consider primarily those requirements from the catalog that were rated to be most important for these stages.

Table 5.1 shows a list of the requirements derived in the expert review, sorted by their mean importance for prototyping tools in very early, early, and middle design stages. The last five entries of this list display the suggested categories, which mean value cannot be compared to the above requirements, since few or no experts gave ratings for the suggested categories.

Ratings were done on a 5-point likert scale, ranging from 1 = “unimportant”, 2 = “rather unimportant”, 3 = “middle important”, 4 = “rather important”, to 5 = “very important”. In the following discussion of most suitable requirements for the user study, primarily the *most important* requirements are considered.

The classification in *most important* and *less important* requirements for the first three development stages was done in two-step proceeding. First, I decided to divide the list into those requirements that ranked 3.3 or better, and those that ranked 3.1 and lower. The value 3.1 is close to the rating value 3, which indicates a *middle importance* of the requirement.

As a second step, I investigated the list of lower rated categories, to make sure that it does not contain requirements that might be rated highly in single phases of regarded timespan. The highest single values were found for the categories *reusable prototypes* and *easy setup and distribution of user tests*, that were both rated 3.4 for middle development stages. I judged the rating of 3.4 in a single phase to be not high enough to justify a categorization into the most important requirement dimensions.

Consequently, the following discussion in Sect. 5.2.3, on whether and how to apply the most important requirements in the user test, the first eight of the properly evaluated requirements listed in Table 5.1 are regarded. As for the suggested categories, which were only rated by a limited number of experts, the categorization described in the paragraph above cannot be applied. Therefore the application of each of the five suggested categories is singularly discussed.

### 5.2.2 Considering the Type of Evaluation Method

The most important requirements for early to middle design stages shown in Table 5.1, can solely be measured in evaluations that survey the practical application of the tools. Especially for the investigation of requirements like the *collocated group work* or the *freedom of creativity and design* a tool application study seems to be the only feasible approach.

Another important factor that has to be taken into account is the timespan sufficient to the evaluations. Creativity is a complex cognitive process that does not necessarily work within the flick of a finger. Creative processes in teams tend to complicate the matter even further. Beside that, the investigation of a prototyping process cannot ignore the analysis of the prototype outcome and testing. In case of the PBP approach, where tests sessions involve a lot of manual manipulation by the design team, a prototype result can only be judged in an observation of the test session.

To compare the application of different prototyping approaches by teams that work for a reasonable period of time, a long-term observation of the tool use in practice is a possible approach. In such a study, a professional mobile app development company could be visited that uses the selected prototyping tools in their design process. The big advantage of this approach is that the design sessions could be observed throughout multiple iterations that might span weeks or even months. Taking this evaluation approach into consideration, I asked some of the experts I got to know during the expert reviews described in Chap. 3. From a couple of them I received positive replies to try out the Blended Prototyping approach in practice.

However, I decided to conduct a comparative user study, rather than evaluating the tool in a long-term real application context. The real-work observation has the big disadvantage of not allowing for the controlling of a number of important factors. For example I hardly would have had the power to dictate, which members

should work in a team, what prototyping tasks the teams must address, or when and for how long which tool must to be applied.

These factors however need to be controlled for a clean comparative evaluation with performance measures. For this reason I decided to conduct a constructed user study with paid test participants that work in allocated groups, on given tasks, for a controlled amount of time, with each of the selected prototyping tools. This allowed for a within-subject evaluation, where each participating group worked sequentially with each tool.

Long-term practice studies are a good and necessary means to observe the prototyping process throughout multiple iteration phases. They are without alternative when it comes to analyzing the long-term use and succession of different prototyping techniques throughout the development cycle. For a comparative evaluation between different prototyping tools with respect to the key performance indices identified above, however, a constructed user-test is to be preferred.

### *5.2.3 Discussing Assessment Methods for Identified Requirements*

In the previous text the choice of the surveyed prototyping tools was explained, important requirements for the comparative evaluation at earlier design stages were identified, and the decision for a controlled user study as the most suitable evaluation method was described. In the following all requirements that were identified to be relevant for very early to middle phases are discussed with respect to their application in the outlined user study approach. This discussion involves the explanation of methods that can be used to assess tools in accordance to the requirements. For those requirements that cannot be investigated in the planned user-study an analysis is conducted on how the surveyed prototyping tools fulfill the requirements on the basis of objective factors.

The discussion regards all 8 categories that were evaluated to be at least of rather high importance for very early to middle design stages. Since no comparable ratings for the relevance of the categories suggested by experts is available, the discussion furthermore considers each of these requirements as candidates.

1. **Getting Quick Prototypes**

Many authors in related work concentrate their efforts on the improvement of speed in single tasks of the prototyping process [1, 15, 154]. The success of such improvements is measured in comparative user tests, where the time needed to fulfill a certain interaction task is recorded.

The procedure of task time measurement can be used to investigate the whole prototyping process as well. However, here one issue needs to be reflected upon:

The time it takes a design team to complete their solution does not solely depend on the characteristics of the used tool, but on the character of the solution they find

and produce. More complex or detailed solutions will naturally take more time to be implemented into a prototype than simpler ones.

It is possible, to exclude creative aspects from a given prototyping task to a large extent, for example by requesting participants to implement a prototype according to presented app screenshots. This way, the role of creativity in the solution process is reduced as far as possible and the prototype results are much easier to compare.

However, this reduction of the creative complexity of the design task makes it impossible to investigate, to which degree a design tool is able to benefit the ideation process.

The process of developing an idea into a prototype can be illustrated in three sequential steps: *Getting the idea* regards the first step, where an interface idea comes to a designer's mind. It is then *substantiated* in a way, where ways are explored on how to put the idea into a suitable design. Here, the idea is expressed in a suitable design, which regards the user interaction within the interface. For this, usually simple hand sketches are used. Now, the tools of the applied prototyping approach are used, to *transfer* the concrete interface idea into the prototype. The steps of this process are iterative and are not always gone through sequentially. In the process of putting an idea into reality, oftentimes issues occur that need initial reconsolidation. When it comes to testing prototyping tools within user tasks, the first two steps of this process are hard to control. They are highly dependent on the test subjects' initial idea and motivation to transfer into a detailed prototype.

As displayed repeatedly in Chap. 2, different authors [31, 86, 138, 140, 154] point out that tools have a big impact on the ideation and creativity process. Many tools aim to merge the steps of ideation and production to facilitate an improved hands-on experience with ideas already in the process of their development. Beside that, the expert survey described in Chap. 3 identifies the 'Support of creativity and design' as a key requirement that is of high importance for a development tool especially in early design phases. Therefore, ignoring ideation and creative aspects of the design process, to receive better comparable time measurements, does not seem to be the right approach.

Measuring the tools' ability to deliver quick prototypes should therefore be based on user tests with design tasks that include creative aspects. Here, the differences in the produced prototype results can be used as a performance measure, if the time granted to the participants for the design task is controlled. This way, the measurement of the tool's ability to deliver quick prototypes is expressed in analyzing the tools' time efficiency.

Figure 5.1 shows the connection of the two major factors that determine the prototype result of such a user test. It assumes that the created prototype is dependent on the individual performance of the test subjects working on the task, as well as the tool ability. Since the goal of the experiment is to solely measure the second of these factors, the subjects' performance should be leveled out. Therefore, the grouping of teams that perform the study should be done under the premise to establish a comparable skill level between the participating teams. Besides that, a within-subject experiment design, where each team tests each of the methods, helps to balance out the individual effects.

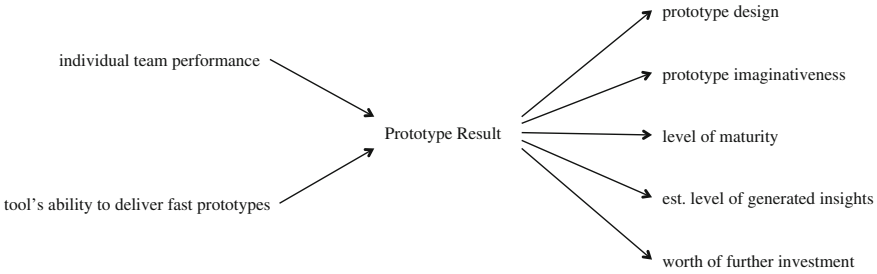


**Fig. 5.1** Measuring the time-efficiency of a prototyping approach on basis of delivered results

On the basis of the assumption that the individual team performance can be balanced out in that way, differences in the prototype result are able to show the direct influence on the *time-efficiency* of a prototyping tool. In other words, the tool's time-efficiency can be determined from the overall quality of the prototype result. The assessment of the quality of a prototype is best done with experts with experience in mobile app design and development. On the basis of their personal experience they are able to judge the prototypes' success to fulfill different aspects of prototype goals. As displayed in Fig. 5.1, the subscales *design*, *imaginativeness*, *level of maturity*, *level of insights*, *worth of promotion* are used to differentiate the question on the overall prototype quality. The selection of theses subscales is based on the prototype information goals described above in Sect. 2.1.2. Consequently, in this proceeding not the *quickness* but the *time efficiency* of the prototyping tools is measured.

For the reason to regard the ideation process in the investigation of the prototyping tools' time-efficiency, the analysis of the user test described in this chapter will be based on judgments derived from expert ratings.

2. **Collocated Group Work**

The investigation of the work with the involvement of computer systems is a complex and distinct field of research. There is a lively debate about the evaluation of collaborative work in the CSCW (computer supported collaborative work) community.

On the one hand, the discussion regards objective measures, which are used to judge the collaborative performance of study groups by an external rater on the basis of observations. On the other hand, self-reporting instruments are discussed, which are questionnaires or interview guidelines to generate data on the subjective opinion of subjects that participated in a group work task.

The objective approaches to assess collaborative work distinguish themselves in the degree to what the rater has to interpret the observations. Approaches with a low depth of interpretation reduce the responsibility of the rater to count the occurrences on specific effects that indicate collaborative interaction. Related work shows a number of such approaches, however, they usually focus the evaluation of single aspects of collaborative work. For example, Wallace et al. [155] or Takano [147]

investigate metrics that are primarily related to aspects of *communication*, Hornecker et al. [64] investigate *awareness*, and Gutwin and Greenberg [55] focus on the *coordination* between the collaborators.

In an attempt to combine different subscale criteria to a general assessment framework, I used a taxonomy by Dickinson and McIntyre [43], which explains collaborative work as a combination of the different factors named above. The taxonomy was used to allocate different related work to the factors, which provided metrics to measure single aspects of collaboration. The result of this work was a coding-scheme that broke down the investigation of collaborative work to a long list of countable events.

This scheme was tested and adjusted in four iterations with test raters that applied the tool in assessing test videos of design sessions. Although the scheme was iteratively refined, no sufficient consistency between the test ratings could be yielded. For this reason, I eventually discarded the approach. The last version of the development can be found as Appendix B.1 attached to this work.

An existing evaluation scheme can be found in a tool called OTAS. The tool is widely used, though not in the field of human computer interaction but in medical research. OTAS, an abbreviation for *Observational Teamwork Assessment for Surgery*, is an instrument to assess the quality of teamwork in medical surgery. OTAS follows an approach, where more responsibility is put into the hands of the coders. Here, they do their ratings in the aggregated categories *Team Orientation*, *Team Leadership*, *Communication*, *Awareness & Monitoring*, *Feedback Behavior*, *Backup Behavior*, and *Coordination*. An OTAS coder is to be trained carefully to get a profound understanding on the meaning of the single categories. In the actual ratings, she judges a group work session with solely one value for each of the subscales. More exact reasons for the single ratings can therefore not be reconstructed after the assessments.

The OTAS scheme asses team work in nearly the same categories like the reference model of Dickinson and McIntyre [43], which I used as a basis in my attempt to create a self made objective coding-scheme (see Appendix B.2). The metric is well applicable in post-session video analysis and therefore qualifies for an application in the analysis of the comparative prototype tool evaluation.

Different subjective self-reported measures on teamwork are found in the field of CSCW research, however, mostly they are targeted at specific collaborative tasks. Sauppé and Mutlu [129] developed a well adapted questionnaire that can be used to assess different forms of collaborative work. The approach analyzes collaborative work with a questionnaire that records in different dimensions, which include aspects on the interpersonal relationship among the members participating in the teamwork as well. The result of teamwork can be highly affected by such factors. If persons who do not like each other at all have to work collaboratively in a teamwork task this oftentimes reduces their motivation, which will reduce the quality of the generated outcome [129]. In detail, the questionnaire surveys *teamwork* (7 items, Cronbach's alpha = 0.811) and *collaborativeness* (4 items, Conbach's alpha = 0.701), along with factors on the interpersonal relationship between the team mates, which *rapport* (24 items, Cronbach's alpha = 0.920), *empathic*

*concern* (7 items, Cronbach's alpha = 0.764), *perspective-taking* (7 items, Cronbach's alpha = 0.820), *interpersonal solidarity* (14 items, Cronbach's alpha = 0.774), and *homophily* (9 items, Cronbach's alpha = 0.812).

The questionnaire by Sauppé and Mutlu provides a good approach to measure the self reported group work in the comparative tool evaluation. Its interpersonal factors can be used to make sure that interpersonal antipathies do not falsify the measured prototype outcomes.

3. **Freedom of Creativity and Design**

To measure the capability of a tool to facilitate *freedom of creativity and design* in its work process, an investigation of the idea generation process or an analysis of the level of creativity in the design outcome could be possible.

As creativity is a high-level cognitive process [30], for the analysis of the creative process cognitive models would have to be applied that are able to classify and measure the creative thinking of designers. Such a level of understanding of the ideation process does not exist [85, 158]. Of course, it could be possible to ask participants of a design process to protocol their creative thinking in a think-aloud method. However, as Dorst and Cross [46] point out commonly agreed upon methods to analyze such protocol data does not exist. Therefore, I discarded the idea to measure *freedom of creativity and design* from an analysis of the design process.

For the analysis of creativity on the basis of the resulted outcome, different approaches exist. Hilliges et al. [60] analyze the creativeness in the use of a brainstorming application by counting the number of ideas generated by the users, as well as the quality of the ideas, rated on subjective judgment. However, the sheer number of created ideas might not be a well applicable approach for the planned study. Where it is the goal of a brainstorming process to create as many ideas as possible, in the prototype development task given in the planned study, the design teams are asked to implement the ideas in a given task, wherefore a brainstorming phase happens but is limited in time. A widely used approach by Shah et al. [133] suggests the measurement of the ideation novelty. This novelty score is calculated on basis of the number of ideas a certain design group generated as the result of a creative process, and puts it into relation of the number of occurrences of the same idea in the work of other groups.

4. **Simultaneous Tests of Different Ideas**

The simultaneous test of different ideas has different advantages. Parallel tests provide reference points, both, for the rating of an approach and the interpretation of the measured performance. Furthermore, e.g. Snyder [140] underlines that the parallel development and testing of different design alternatives at the same time, makes it easier for design teams to be ready to discard ideas that do not perform as well as planned. Moreover, Dow et al. [47] report findings, that developing for simultaneous tests of different ideas encourages more dissimilar designs and in fact produces a higher quality prototype outcome. Besides, they found that a simultaneous design and test of different prototyping alternatives yielded in increased

self-efficacy of the development teams, whereas those who tested their prototypes in a serial way one after another were more likely to get frustrated.

The ability of a tool to promote the simultaneous testing of different ideas depends on different factors. The speed with which prototypes can be created and altered is certainly an important factor.

Furthermore, the prototype test processes should enable an easy setup of comparative tests. The testing of the factor in a user test as demonstrated by Dow et al. is possible, who hired experts to provide initial feedback on the different prototype results tested. In the conceived comparative study this proceeding would prolong the study execution time to an unacceptable extent.

In the outlined experiment, user groups will have to work under a time-constraint. This means, that the users will have to take the decision whether to develop one more advanced idea into a prototype, or to develop different approaches of less complexity. In the tests it will be left open to the participants, which tactic they like to choose. Explicitly asking users to develop alternative approaches would reduce the single prototypes functionality, wherefore other metrics like the time-efficiency are hard to measure.

5. **Support of Expert Reviews**

Expert reviews, like the cognitive walkthrough or heuristic evaluation, are analytical inspection methods where domain experts apply a pre-defined systematic to evaluate a product.

For judging the ability of a tool to support expert reviews, data needs to be gathered from applying the development tool and its delivered prototypes in expert reviews. As metrics for the success of the tool, standard usability questionnaires can be applied, in which the experts can report their experience with the applied prototyping method. An objective figure could be to measure the time needed to conduct a certain heuristic evaluation. As a measure for the overall success of an expert review, another important test criteria should regard the number of usability problems identified by the expert reviews.

As pointed out above in Sect. 5.2.2 I decided to conduct the comparative evaluation within a user study, where participants fulfill creative prototyping tasks in teamwork. Furthermore, I decided to request the test participants to conduct a prototype test session, to provide a basis for the later analysis of the prototype results. Especially paper-prototypes can hardly be judged without the observation of their application in a test condition. In addition to this, as a part of evaluating the prototyping-tools, expert reviews could be conducted to analyze the tools' performance for the support of *expert reviews*.

However, conducting expert review requires profound skills and planning: Expert reviews should be well adjusted to the specific application, suiting heuristics have to be identified, or exemplar person as need to be designed. Furthermore, performing the expert review has to follow certain sets of strict rules that the experts applying the methods need to be well aware of.

The decision made above in Sect. 5.2.3 not to evaluate the prototyping tools in a professional surrounding has the consequence that the level of expertise that the test participants will provide will be limited.

As a consequence of the decision made above in Sect. 5.2.3, not to evaluate the prototyping tools in the context of a professional company, it is hard to establish the level of expertise of the participants of the comparative user-study at an expert level. Paying experts for the participation of long enduring cross tool evaluations exceeds the financial capabilities of my research funds. Therefore, I discarded the idea to include expert reviews as a part of the comparative user-study.

However, considering the nature of the identified prototyping methods, a number of clues regarding their performance in expert reviews can be identified. In expert reviews, paper-based prototyping will likely have its strength in providing the design as physical content. Dealing with paper to organize information comes natural: sheets can be grouped paper piles to segment connected aspects of the user interface, or they can be pinned and organized on large walls to provide a better overview of the interface as a whole [86]. A clear disadvantage of the PBP approach lays in the fact, that the user interface 'cannot speak for itself' without the presence of someone who knows about the interface intentions. Therefore, expert reviews with paper-based prototypes create more effort, and external experts are not able to conduct a review on their own.

In contrast to that, a user or an expert uses digital prototypes independently. Therefore prototypes created by Axure or Blended Prototyping can be explored and evaluated by experts without the presence of development team members that explain the prototype. Moreover, since the prototype runs directly on the target device, expert tests in the wild are made possible that might help to identify usability problems connected to mobile use contexts.

Axure prototypes can be printed out onto paper, where after they provide the same materiality of paper prototypes and therefore adapt the described advantages of a direct physical manipulation. However, in their printed version, Axure prototypes fail to explain their functioning.

Using an overhead projected tabletop setup and physical paper, Blended Prototyping combines the advantages of the low- and high-fidelity techniques pointed out above. Here, the overhead-projection of the tabletop computing environment is used to highlight functional aspects regarding the paper screens in the spacious physical world of a table surface. This aspect might play out its strength in the context of expert reviews as well.

6. **Support of Design Reviews**

Testing a prototyping tool's ability to support design reviews can only be tested in the application of the tool or its results in a design review. As pointed out earlier in Chap. 3, design reviews are primarily an instrument for the communication between the supplier and customer of a software development project. Whether or not such communication works well, is to a large extent dependent on the

participants' subjective perception. Hence, the assessment of this requirement can only be done with real design review praxis tests and subjective quality questionnaires.

The choice of the right tool to conduct a design review should not least be depending on the specific customer. Some customers might feel irritated by the simplicity and low development effort of a paper prototype, for others, who are accustomed to the approach, the approach might be well suitable in early design stages. After all, the use of prototypes in customer relations should always make clear, which aspects of the prototype are implemented to what extend.

7. **Tests in the Real Use Contexts**

The requirement, that a prototyping tool should support testing of prototypes in the real use contexts of the app, does not depend on the design process, but on the character of the prototype result. Therefore, this requirement is not tested within the tool-application study discussed in this chapter.

Different authors point out limitations of using the paper-based prototyping approach in user tests in the field. Problems are mainly experienced in the observation of the test user and her input commands, wherefore the manipulation of the presented prototype content cannot be performed appropriately. Beside that, mobile paper-based tests tend to fail to create a use contexts that is perceived natural by the test users, when they are followed by a group of observers trying to sneak over their shoulder on every step that they take. Moreover, long-term prototype studies with paper-based prototypes are hardly feasible. Such studies are however often very relevant, since many mobile applications are used as a service: for a limited session time only, but many times a day.

In contrast to that, Axure produces digital prototypes that can be ran and tested directly on mobile devices. Therefore, test users can use and explore such prototypes in the mobile context just by themselves. Even long-term tests can be easily conducted, where a prototype is used every now and then but for a couple of days.

Unfortunately, Axure provides no mechanisms to automatically track and save data on the prototype usage. This means that the feedback generated with Axure prototypes is limited to the subjective reports of users after the testing. Statistical analysis of quantitative data cannot be conducted. However, it is often of interest for the designers to learn for example how often users accidently chose wrong paths in the user interfaces, or how long they stayed on certain screens. The missing of a logging feature is however not a problem inherent to the prototyping approach, but is rather a feature that not provided in the current implementation.

Just like Axure, the Blended Prototyping approach produces digital prototypes that are tested directly on the mobile device. Unlike Axure, however, it implements a logging functionality that tracks each touch command of a user during a test. Therefore, the exact interaction of a user with the prototype can be reproduced and analyzed after the test. Moreover, the approach facilitates mechanisms that put developers in a position were they can change the prototype assigned to a test user,

literally seconds before the testing. This way, changes in the prototype can be regarded in the testing without the users' notice.

8. **Reusable Programming Code**

While the graphical design of user interfaces is nowadays typically created with interface building tools that use a markup language, the behavior of a software prototype is determined with programming code. The reusability of programming code in a prototyping approach can refer to an instant reusability from one iteration cycle to another, or to a long-term reusability in the programming code of the later product.

In the planned user test, the reusability of code can hardly be investigated, since the number of iterations in the prototyping process will be too limited.

The paper-based prototyping approach does not use programming code in a typical way. In fact, programming languages that are compiled to machine code are not used, since the 'machine' that is used in paper-based prototype tests is actually a human being. However, in preparation of paper-based tests, written guidelines are created that help the team member playing the role of the computer to respond correctly to the possible test user input. Regarding such written scripts as the programming code of the paper-based prototyping technique, allows for an analysis about whether or not the programming code of the PBP approach is reusable.

Scripts used for the implementation of PBP tests do not regard each functional aspect of the prototype equally, but are rather a collection of personal reminder notes that help with the execution of particularly complicated points in the prototype. From one iteration cycle to another, such notes could be easily reused in the parts that stay valid in the next iteration. However, for two reasons, such notes have little to no value for the programming of the later product. First, the scripts do not necessarily describe the interface function entirely, but are rather collections of personal reminder notes that help with the execution of particularly complicated points in the prototype. Second, since such scripts do not follow a formal standard, they are likely to include notes that are incomprehensible for people other than the author.

In the prototyping tool Axure, behavioral aspects are programmed into the prototype with short code snippets, the execution of which is linked to specific interface events like button clicks. Such programming code can be seamlessly adjusted and reused from one iteration cycle to the other. The situation is different when Axure code is reused in the later product, since Axure does not use a native programming language, but a simplified Axure specific programming technique. Hence, behavioral aspects of Axure prototypes have to be reprogrammed to a large extent in the production of the product.

As displayed detailed in Sect. 4.2.3, Blended Prototyping uses Java classes to determine the prototype behavior. Since Java is the native programming language to the Android platform, code created in the Blended Prototyping process can be principally directly reused in the programming of Android applications. As discussed in Sect. 4.2, to simplify the production of fast prototype results, Blended

Prototyping does not cover each Android specific programming mechanism. For example, specific questions on UI fragments, resource management, or access rights cannot be explored and solved in the Blended Prototyping process. Apart from that, however, Blended Prototyping can cover even more complex programming scenarios, which might occur in developing the app's interplay with servers, in apps that are using the devices' sensor infrastructure, or to try out and develop even complex algorithms.

9. **Suggested Categories**

The requirement of an *open source availability* of the tool is a bipolar measure that can be answered with a simple yes or no. The Blended Prototyping approach is not available open source yet, but is planned to be published under open source licenses after additional code reviews. As for the paper-based approach no license rights at all exist for the method. The Axure software is a professional commercial tool distributed closed source.

The *compatibility of the tool with different platforms* can be answered easily for the three tools as well. Axure is available for Microsoft Windows and Apple Mac OS, but not for Linux operating systems. Blended Prototyping runs inside the Java VM, which is known for its compatibility with all major operating systems. Adjustments to hardware drivers for employed cameras might be necessary, however, as displayed above in Sect. 4.3, the camera control algorithms use open-source techniques that are available for Linux, Mac OS, and Windows systems. The paper-based prototyping approach is not bound to a computer, therefore the requirement of compatibility to different platforms cannot be applied.

*Tutorials and help for the tool* are supplied for Axure within the software, with forums on the publisher's web page, and instructive videos available on YouTube. Since paper-based prototyping is a common method, there is no official supplier providing documentation. Nevertheless, detailed descriptions of the method exist for example in the book of Snyder [140] that was referred to oftentimes in this work, or in countless online tutorials. Currently, the help and documentation for Blended Prototyping solely exist in a preliminary state. As a consequence, the *tutorials and help for the tools* can currently hardly be compared.

As for the suggested requirement *fun to use the tool*, I decided to broaden the scope of the figure and measure the *user-experience*. The ISO standard 9241-210 defines the user-experience as "a person's perceptions and responses that result from the user or anticipated use of a product, system ore service". Therefore the figure broadens the focus of the *Fun of tool use* and includes further criteria that might be very relevant in using productive creative tools. A positive user experience is as well capable to promote creative work [50].

The suggested requirements dimensions *usability of the tool itself* and *user-experience* are closely related. Both dimensions are of key importance for the research field of human computer interaction, are handled in the ISO 9241 standard and are discussed in detail in countless scientific and practitioner publications. Melting down the meaning of the terms into one sentence, *usability* refers to the

ease of use and learnability of a product [108], whereas the *user-experience* concentrates on the emotions and attitudes a user-experiences in the interaction with a product [112, 113]. For both fields numerous evaluation methods are described. Its summary goes far beyond the scope of this work.

The metric I decided to use for the assessment of both, the *usability* and *user-experience* of the surveyed tools, is the self reported questionnaire AttrakDiff by Hassenzahl et al. [58]. The questionnaire combines the self-reported assessment of usability and user-experience aspects of a tool on the scales *pragmatic quality*, *hedonic quality* (in the subscales *identification* and *stimulation*), and *attractiveness*. The questionnaire uses a set of 28 semantic differentials that assess different product properties with antonyms like "good-bad" or "confusing-clear". The sub scales of the questionnaire are determined from different sets of these differentials. The pragmatic quality of AttrakDiff can be translated to a self-reported *usability* measure. The hedonic quality, which is further divided into the subscales identification and stimulation, are related to the perceived *user-experience*. The subscale attractiveness gives a summary of the overall quality of the surveyed tool.

## 5.3 Conducting the Comparative Study

As explained above, the evaluation measures effects that the use of different mobile app development tools might have on the prototyping process and its results. The study investigates how test users in groups use the three different prototyping tools Blended Prototyping, Paper-based Prototyping, and the software Axure to create prototypes for different creative tasks. To measure effects on factors like collaboration and ideation processes, groups have to be involved in creative teamwork sessions, where they work freely and independently on the app prototyping tasks. Establishing a suitable surrounding for this creative group work to develop has to take into account a number of different aspects. How do the creative tasks need to be constructed? How much time is necessary, to allow for group ideation sessions that deliver prototype results? Which prerequisites should the test subjects fulfill and how should they be grouped in teams?

This section first explains the study design that was used for the comparative evaluation along with a number of design decisions that were made to produce creative teamwork prototyping sessions. Then the results of the evaluation are presented, after which follows a discussion of the study results.

The development state of the Blended Prototyping approach when the user tests were conducted was as described in Chap. 4, but excluding the low-tech definition methods for the design-tool. Unfortunately, at that stage they did not provide a high enough reliability to be used in a productive team scenario. Therefore, the users had to control the tabletop environment with the described control app, running on a tablet device.

### 5.3.1 Study Objectives

The main objective of the user study was to compare the performance of the Blended Prototyping approach with the tools of Paper-Based Prototyping and Axure. As criteria for the evaluation serve the requirements identified above in Sect. 5.2. This regards the requirements about the *time-efficiency in the prototyping, collocated group work, freedom of creativity and design, usability*, and *user-experience*. An extensive discussion on why the following methods to assess these requirements were selected is found above in Sect. 5.1. To provide a better overview, the assessment methods are summarized here.

To include factors of the ideation process into the measurement, the requirement of *time-efficiency in the prototyping* is not measured with a stopwatch, but on the basis on the delivered prototype results. The rating of the results is done by external experts, which do their rating on the five scales identified above.

The performance index *collocated group work* is assessed in two different ways. For a self reported subjective perception of the performed group work the participants report on a collaboration questionnaire developed by Sauppé and Mutlu [129]. In addition to that, as an external measure of the group work the rating schema OTAS is applied to rate recorded videos of all the study's productive prototyping sessions.

The *freedom of creativity and design* is measured as a part of the same rating of the created prototype result, which is applied to measure time-efficiency of the prototyping tools. For the rating of the performance to create free creative design processes the scales *look* and *inventiveness* are used. Implicitly information can be gained in addition from the measured results for *generated insights* and the likeliness for a *further investment*.

In addition to that, the method introduced by Shah et al. [133] is used to assess the *novelty* of prototype results, as an important factor of their ideation effectiveness. This approach measures the inventiveness of prototype results by comparing the appearance and scarcity of their features, created for task specific categories.

The *usability* as well as the *user-experience* of the tools are both measured with the AttrakDiff questionnaire. As explained above, the *usability* is measured with the questionnaire's sub scale on pragmatic quality, whereas the *user-experience* is investigated with the two sub scales that concern the measured hedonic quality.

### 5.3.2 Study Design

#### 5.3.2.1 Test-Subject Acquisition, Pre-questionnaires and Formation of Groups

Test subjects were mainly recruited from the university context. Bulletins were put up widely across the campuses of the Berlin University of the Arts (UdK Berlin)

and the Technical University of Berlin (TU Berlin). In addition to that, an online test subject database of the former DFG graduate school *prometei* was used.

Though the use of a specific prototyping tool can have a strong effect on the prototyping process and outcome, there are other factors to this result as well. To actually measure the effect of the tool use, other factors with similar effects must be controlled. One important of such factors is the individual level of skills of the individuals forming the design group. To control this effect, study applicants went through a pre-selection process, where their individual capabilities were assessed. The formation of the groups then used the results from this pre-questionnaire as a basis for an equalization of the average group capabilities. For the pre-assessment three questionnaires were used, one containing general questions, a questionnaire to assess the creative capabilities, and one to get an idea of the basic programming knowledge of the applicants in the programming language Java.

The questionnaire applied to measure the applicants creative potential was based on the *Creativity Achievement Questionnaire* (*CAQ*) developed by Carson et al. [33]. The CAQ is a widely used tool in its domain, which is based on self-reporting experience levels in a collection of 14 different creative domains. However, pretests of the CAQ with friends and colleagues, for whose creative potential I had previous experience knowledge, showed that the original question set of the CAQ was not well applicable for the planned pretests.

The major problem about the standard CAQ laid in the differentiation of the answer options for each category of creativity. For example the options offered for the category Music are as follows:

0. I have no training or recognized talent in this area
1. I play one or more musical instruments proficiently
2. I have played wit a recognized orchestra or band
3. I have composed one original piece of music
4. My musical talent has been critiqued in a local publication
5. My composition has been recorded
6. Recordings of my compositions have been sold publicly
7. My compositions have been critiqued in a national publication

This differentiation makes clear that the CAQ addresses the assessment of a professional creative background. In the conducted user study, however, not domain experts, but mainly ordinary students were addressed as participants. Such applicants would rather score between 0 and 2 of the scales, which makes a suitable differentiation impossible. For this reason, I decided to adjust the scale to a level of less expertise, as shown below, again for the category Music:

0. I have no training or talent in this area
1. I play music from time to time in my spare time
2. I play music regularly in my spare time
3. I took music lessons (beside school) for a longer period of time
4. I have played music in front of a public audience
5. I already earned money for playing music

6. Music is my profession
7. I earned awards for my music

A repeated test of the questionnaire with the adjusted scales, with same persons as the first pre-test, delivered much better differentiated results between the surveyed. A second adjustment of the standard CAQ was done with regard to the applied categories of creativity. In the original CAQ version, these regarded *visual arts (painting, sculpture)*, *Music*, *Dance*, *Architectural Design*, *Creative Writing*, *Humor*, *Inventions, Scientific Discovery*, *Theater and Film*, and *Culinary Art*. To these categories I added two new ones, that are closer related to the tasks in the user study: *Application- and web design* and *Programming*. The adapted CAQ used in the group allocation process will be further referred to as the *custom CAQ*, abbreviated with *cCAQ*.

To acquire previous knowledge about the applicants programming skills, a custom Java questionnaire was designed and tested. Standardized questionnaires can be found but are usually for certification purposes, like in the Oracle Java SE Programmer Certification Program.[1] Standard tests that address the assessment of beginner programming level in Java do not exist. That is why I formulated 10 simple Java questions that assess the familiarity with the Java programming language at a very basic level. The questions were iteratively tested and adjusted with friends and colleagues of different skill levels. All pre-questionnaires are attached to this work as appendices Appendix B.3.1 to Appendix B.3.3.

***Formation of Groups***

The cCAQ is a self-reporting instrument that asks for experience levels in different creativity related domains. It relies on truthful answers and has no means to indicate cheating. The created Java-questionnaire asks about programming knowledge questions, however, a clever cheater might easily identify the correct answers with a short internet search.

Therefore, candidates were informed that the results of the pre-questionnaires were not deciding about whether or not they were allowed to participate in the study, but were used as a basis to sort them into different groups. They were further asked to answer the questions truthfully, to give the answers just by themselves, and to not search for answers they do not know online or in other sources.

The group members for a test session were chosen on the basis of the pre-questionnaire results. As a premise, each group was assigned with at least one member who was rated specifically high for the cCAQ (above 80%), and at least one different member who was rated specifically high for the Java questionnaire (above 80%). The fact that three sessions were held in parallel made it easier to assign applicants to suitable groups at available times.

[1] https://education.oracle.com (last accessed 11th April 2016).

***Participants***

All together 36 test subjects participated in the study (16 female, 20 male), aged 26.5 years (SE = 1.07). A majority of the participants were students (27), the others worked in regular employment. The average score of a group for the cCAQ was 18.14 (SE = 1.15), for the JavaQuest a mean of 5.25 (SE = 0.312) was produced.

As described detailed in the following Sect. 5.3.1.3, the time effort it took the subjects to participate in the test was approximately 12 h. Due to this long working time, the participants had to come to the testing facilities at two days. For their high efforts, the test-subjects were paid the comparatively high compensation of 120 € for their participation.

#### 5.3.2.2 Test Design

Beside this pre-assessment based group formation, a within-subject study design was chosen to balance out effects of individual group performance. This means that each group used all three prototyping tools sequentially, where the order of tool use was balanced between the groups. The task assigned to a tool-use varied: a group never worked on the same task twice, the order in which the tasks were given was randomized. The within-subject study-design is shown in Table 5.2.

For a better understanding of the terms used in the study discussion, the following terms are defined: A *cohort* is constituted of the groups that participated at the same time in parallel in the user tests. The term *group walkthrough* refers to the complete participation in all three tasks of a single group (a row in the table above). An *assignment* includes a certain given task that is to be processed at a specific tool.

#### 5.3.2.3 Testing Process

The testing followed a fixed procedure that guided the participants through an introductory presentation, the three productive sessions, and a short conclusive

**Table 5.2** Schematic study design

| | time-slot 1 | time-slot 2 | time-slot 3 |
|---|---|---|---|
| *Cohort 1* | | | |
| Group 1 | Ax_T1 | BP_T3 | Pb_T2 |
| Group 2 | BP_T3 | Pb_T1 | Ax_T2 |
| Group 3 | Pb_T1 | Ax_T2 | BP_T3 |
| *Cohort 2* | | | |
| Group 1 | Ax_T1 | Pb_T3 | BP_T2 |
| Group 2 | BP_T3 | Ax_T1 | Pb_T2 |
| Group 3 | Pb_T1 | BP_T2 | Ax_T3 |
| | …<br>… | …<br>… | …<br>… |

| | |
|---|---|
| ~ 30 mins | explanation of the tool, tryout task |
| 5 mins | pause |
| ~ 5 mins | explanation of the design task |
| 90 mins | work on the design task |
| 5 mins | pause |
| ~ 20 mins | task result presentation |
| ~ 15 mins | answering the questionnaires |
| ~ 170 mins | sum |

**Fig. 5.2** Calculation of the time needed for an assignment

questionnaire. As a first step for each cohort, I gave an opening presentation. Here, a written script was followed to welcome the participants and present them the main study objectives. Further, I once again explained to them the structure and time slots of each of the three productive sessions and introduced them to the teams they would work in. All participants were informed about their personal rights during the study, about the payment process, and the different ways data was collected in the study were pointed out. The participants then gave a written agreement to the study's terms and its mechanisms to collect and process data. After all occurring questions had been answered, the members of a design group were asked to come together for 15 min to get to know each other.

***Productive Sessions***

After the introductory session, the groups started the tool related productive sessions that were structured in the way pictured in Fig. 5.2. Most of the given time-values are estimates and marked with a tilde. These phases were shortened or lengthened, depending on the single situation. E.g. the explanation of the Axure software took usually longer than this into the paper-based prototyping approach. No such time adjustments were made for the design task, where an equal task time was necessary to produce comparable results.

The time a group had to work on a given task was fixed to 90 min. During that time, initial ideas were found and discussed, put into suitable designs, and developed into a presentable prototype. However, before the 90 min prototyping session, time was needed for the introduction of the tool use and a try-out session. After the prototyping session, time was needed for the test of the prototype and to fill out the AttrakDiff and Collaboration questionnaires. The total time of one tool-test procedure added up to approximately 3 h.

Due to the within-subject study design, each group was involved in three design sessions, which lead to a total time effort of 9 h for each participant. I discarded the idea to run all three design sessions within a day, since breaks and lunch time would have added up the span necessary for the participation to 10–11 h. Asking test subjects to work such a long time can easily result in participants' tiredness or even annoyance. Both effects can jeopardize the creative outcome of sessions later in the day immensely. I therefore decided to stretch the participation to two days. To find time slots suitable for the participants, the user tests therefore primarily took place

at weekends. Concerns about participants showing up at the first, but not at the second day, luckily proved to be unjustified.

A reduction of the time for one tool-use would have resulted in a reduction of the time available for the design task. However, shorter time would have led to less elaborated prototype outcomes, where effects of the prototyping-methods on the results were hard to differentiate. On the other hand, a much longer time would have stretched the time needed for the study participation to an extent, where it was unlikely to find enough test subjects.

For a number of reasons I decided to host up to three test runs in parallel. Each design tool was set up in a different room, where I, or one of the assisting student workers, supervised the assignment. This not just saved a lot of time in the overall implementation of the user study, but made it much easier to put the participants into suitable groups. All participants were instructed not to talk with members of the other groups; neither about the tasks and solutions they worked on, nor about the tools they used.

#### *Explanation of the Tool and Tryout Task*

Each productive session started with a phase, where the idea and most important functions of a prototyping approach were explained to the design groups. When the team had no further questions about the approach, they were asked to test their understanding of the tool in applying it in a short non-creative example task, which was the same for every tool-use. It asked to create a user-interface that consisted out of three screens. The first of these screens was meant to provide a short user dialog, the second to have a blue, and third to have a pink background color. The dialog of the first screen was meant to ask the user to select their gender. In reaction to the user's selection, the user interface prototype was meant to switch either to the second blue, or the third pink screen. The tryout task was completed, when the group managed to produce a testable prototype that provided the demanded design and behavior.

In successfully completing the sample task, the test group proved a basic level of understanding of the prototyping approach. They were able to create different screens that express certain states of an app idea. Furthermore, they had to deploy some basic user controls in their prototype, to provide the function for the gender selection and screen change. The users were free to choose what controls they used to realize the app.

#### *Explanation of the Design Task*

After the design task was successfully completed and all questions regarding the tool use were answered, the design task was handed out and read to the test group and all questions regarding its understanding were clarified.

In the case that the design group went through its first productive session, some additional general aspects were explained. The groups were instructed to produce an app prototype, which design and function they could freely decide upon. It was further described to the group, how the later prototype result presentation session would take place. This regarded the ways the group could intervene in the testing

session, as well as their decision, what kind of stakeholder they want the prototype to present to.

In addition to that, the design groups were introduced to a group work ideation method called *brainwriting*. The method adapts the idea of *brainstorming*, but lets the group members write down their initial ideas before they start to discuss it. This proceeding can help group constellations, where some members feel shy to present and stand for their ideas. Whether or not the group applied the technique, was however left open to them.

***Work on the Design Task***

In the design task, the groups were asked to use a certain tool to generate a testable app prototype in the given time of 90 min. How they structured their work processes, was left open to the single groups, no interventions by the supervisor were made. If questions regarding the tool use occurred in the design task, the test supervisor answered them, especially when they blocked the further progress of the group work. To avoid that time consuming explanations take too much of the creative group work time, in-depth technical questions were only answered if they were part of a previously defined catalog. Here, the 24 pages Axure-Core-Training[2] was used as a reference, to decide which technical questions were answered in each prototyping technique. To allow for the later analysis of design task sessions, each session was recorded on videos.

***Task Result Presentation***

Following the design task sessions, the prototype results were presented to one of the supervisors, who did not attend the group's productive session. At the beginning of each test, the design team was free to describe to which kind of test-user they address their prototype testing. As a consequence, an assisting student worker or me jumped into different roles in testing the prototype. Mostly, we were asked to act as a regular test-user, some others wanted us to act like a potential investor, a colleague, or a good friend they consulted.

In which way the design groups then tested their prototype was left open to them. As discussed in Sect. 2.1, prototypes can serve in many different ways to gain insights in a whole range of aspects. Following this thought, a prototype can be understood as a question that is asked by the developer. How clear this question is, whether or not it leaves room for interpretation, and how focused it is towards a certain topic, should be carefully decided from one prototype to another since it has a big effect on the prototype results. Therefore, the prototype test is an important aspect of the prototyping process as a whole.

[2] http://d3g1p8ush40lh4.cloudfront.net/Tutorials/v7/AxureCoreTraining.pdf (last accessed 11th April 2016).

As mentioned above in Sect. 2.1.2, Snyder [140] describes one big advantage of paper-based prototypes as their ability to be flexibly adjusted in the course of a test session. Therefore it was allowed to design groups to adjust their prototype ideas dynamically within the testing process. For this, all means supported by the prototyping approach were allowed.

When the design group decided, that they did not want to address further questions to the prototype test session, it was asked about the insights they gained from the test. Furthermore, the participants were asked to describe which next steps they would make, if the development of the prototype was to be continued.

***Answering the Questionnaire***

Following the task result presentation and feedback talks, each member of the group filled out a copy of the AttrakDiff [58] and the collaboration questionnaire. The participants were placed far enough apart in the testing room to guarantee that all questions could be answered confidentially. Granting this privacy was especially important in our study, since it involved questions regarding the interpersonal sympathy between the test subjects.

When a design group had finished the use of all three design-tools, an additional final-questionnaire was handed out to the participants (see Appendix B.5). This final questionnaire contained questions on the knowledge the test-subjects had about the prototyping techniques before the participation.

#### 5.3.2.4 Task Design

The study design made it necessary to develop three different tasks, which were comparable in their scope and necessary effort. They had to be described in a simple enough way, to be understood by all participants effortlessly. At the same time they had to address application domains that all participants were familiar with, to avoid the variation of test results due to a lack of a personal involvement to the topic. All participants had a university student background. This common personal background was used as a topic for all tasks, where personas were used that faced a university-context related problem, the app should provide a solution to.

The task descriptions were written in German language. All tasks included a certain set of obvious possible features to provide the groups with a starting point for their ideas. At the same time, the task motivation was written in a form, that was held general enough to leave room for individual interpretation.

All tasks concluded with the same two sentences that explicitly motivated the participants to come up with additional ideas, which they could then implement in the prototype. Furthermore it encouraged the design team to concentrate their efforts on those aspects that stand out for the app idea, and at the same time leave out those, which could be considered standards, like login-dialogs. The full text of each task, in German and English language, is attached to this work in the Appendix B.4.

#### 5.3.2.5 Recorded Data

Data was gathered before the study in online-questionnaires, which measured the applicants potential creativity (cCAQ) and basic Java programming knowledge (begJava). Furthermore, questionnaires were answered by each participant after each productive prototyping session. Here, the AttrakDiff and collaboration questionnaire by Sauppé and Mutlu [129] were applied. After a participant finished all three test conditions, they had to answer a final questionnaire that addressed the personal experience they might have had with the surveyed tools before participation.

Videos were recorded for all tryout-, design-, and prototype test-sessions. For this, each supervisor was equipped with a GoPro action-cam that is equipped with a very wide angular lens. This lens made it easy to record information about the applied processes, even in free-hand recordings.

#### 5.3.2.6 Data Acquired in Post Analysis

***Judgment by Design Experts***

To gain a well-founded external judgment of the prototype results created in the user-studies, experts with a professional background of at least two years in mobile user interface development and design were asked to rate the prototype results created in the test. To establish a profound level of experience and hence quality of the ratings, as a prerequisite the participating experts needed to have experience with the design, development, and/or evaluation of mobile user interfaces for at least two years. A total number of 6 experts participated in the ratings, of whom nobody knew the Blended Prototyping approach before their participation. They were compensated financially for their participation with a payment of 20 € per hour.

Expert ratings were done on the basis of all content that was produced in the prototyping sessions. First of all this included the videos that were recorded in the prototype test sessions that followed each design session. In addition to that, all physical and digital content created in the prototyping process was accessible to the raters.

The time needed for the judgment of a single prototype result varied strongly, as the size of the different prototype solutions and the length of the recorded test videos differed. Giving the raters a basis for a comparative rating between the prototype solutions, each expert only rated results belonging to one task. The experts each rated an equal share of prototypes from the three development tool conditions.

For privacy reasons in treating the data from the user tests, the expert rating did not take place remotely, but at the facilities of TU-Berlin Quality and Usability-Lab. For reasons of time efficiency, two or three experts were invited simultaneously to the sessions. They did however work independently with no interaction with one

another. A session began with the explanation of the purpose and proceeding of the rating session. The structure of the prototype data was explained, and an introduction on how to access the related digital content was given. Followed by that, the use of the rating scheme was explained and the description on the meaning of the single rating scales was provided. Throughout the whole rating session, me, or one of the student assistants was present to answer questions that might occur.

All participating experts were German native speakers, for that reason the rating scheme was given in German language. The scheme asked for a rating of the prototype result with regard to 5 aspects: its *look* (Erscheinungsbild), its *inventiveness* (Ideenreichtum), its *level of maturity* (Reifegrad), its *insights it would likely generate for the further development* (Erkenntnisse für die weitere Entwicklung), and its *worthiness for a further investment* (Würdigkeit für eine weitere Förderung). Ratings were given on a continuous scale, ranging over 10 cm from *extremely bad* (extrem schlecht) to *extremely well* (extrem gut). Rulers with measures were supplied, for the case that experts wanted to be very precise in their answer.

### 5.3.3 Study Results

First results of the user study described in the following, were discussed in a chapter of a book that resulted from a research collaboration between the *Universität der Künste Berlin* and *Technische Universität Berlin* [12]

(1) **Control of Disturbing Factors**

The measured results in the user studies can be disturbed by different effects, which risk influencing the measurements in an uncontrollable way. One of these factors is the effect of the given tasks on the prototype results. An investigation with a one-way analysis of variances (ANOVA) showed that the given tasks in fact have a significant effect on the AttrakDiff subscales *attractiveness* (means: $Att_{T1} = 4.98$, $Att_{T2} = 4.85$, $Att_{T3} = 5.51$; ANOVA: $F(2,105) = 3.331$, $p = 0.040$) and *hedonic quality-identification* (means: $HQ\text{-}I_{T1} = 4.57$, $HQ\text{-}I_{T2} = 4.71$, $HQ\text{-}I_{T3} = 4.95$; ANOVA: $F(2,105) = 3.171$, $p = 0.046$). However, since the used tasks were allocated equally to the three tools, the effect should be balanced out in the investigation.

As for the interpersonal relation scales, the effect of each sub scale on the resulting prototype results were measured with a one-way ANOVA. Here, no severe effects of the interpersonal climate within a team on the team performance in the task were identified.

(2) **Measured Results**

***Questionnaires on AttrakDiff and Collaboration***

Bar charts for the subscales of the AttrakDiff questionnaire display the mean rating of each of the three tools above in Fig. 5.3. Visual inspection of the charts shows a

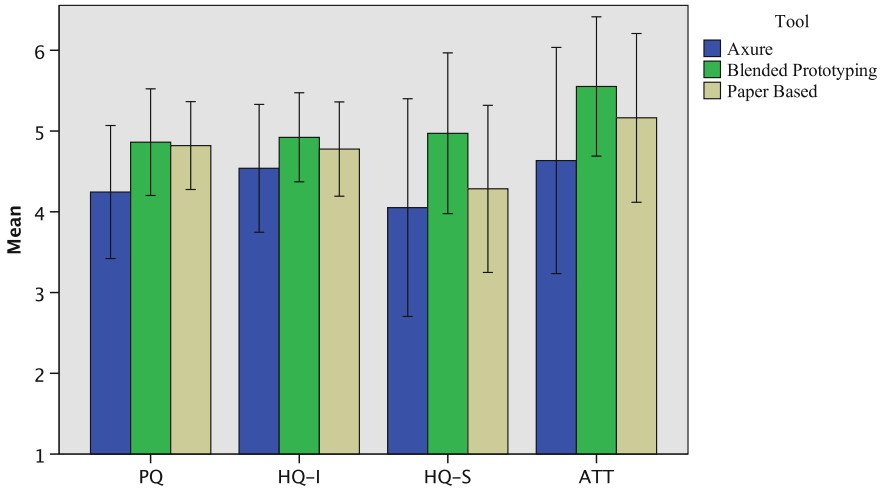


**Fig. 5.3** Boxplots of means for AttrakDiff subscales pragmatic quality (PQ), hedonic quality identification (HQ-I), hedonic quality stimulation (HQ-S), and attractiveness (ATT). (Error-bars ± SD) (scale ranges from 1 to 7)

comparatively low rating for Axure in all of the subscales. The difference between the measured means of Blended Prototyping and Paper-Based prototyping do just differ for the subscales *Hedonic Quality-Stimulation* and *Attractiveness.* For the scales *pragmatic quality* and *hedonic quality-identification* the Blended Prototyping approach just yielded slightly better results.

To investigate the significance of the measured means, data was analyzed using one-way repeated measures analysis of variances (ANOVA) with the factor *tool* having three levels (the experimental groups: Axure, Blended Prototyping, and PBP). All questionnaire subscales were analyzed separately.

For the subscales *pragmatic quality* ($F\ (1.711,\ 59.875) = 9.722$, $p = 0.000$, $\eta^2_{part} = 0.217$, Greenhouse-Geisser corrected) and *hedonic quality-stimulation* ($F\ (1.81,63.353) = 9.443$, $p = 0.000$, $\eta^2_{part} = 0.212$) post hoc Bonferroni comparisons revealed highly significant differences between Blended Prototyping and Axure ($p = 0.001$), as well as between Blended Prototyping and PBP ($p = 0.002$). The difference between PBP and Axure did not prove to be significant.

For the subscale *attractiveness* ($F\ (1.683,\ 58.922) = 6.657$, $p = 0.004$, $\eta^2_{part} = 0.160$), significant differences were found only between Blended Prototyping and Axure. Differences between the other tools did not prove to be significant.

Furthermore, a statistical trend was found ($F\ (1.52, 53.467) = 2.973$, $p = 0.073$, $\eta^2_{part} = 0.078$), investigating the differences in the means measured for Blended Prototyping and Axure.

The second group of bar charts above in Fig. 5.4 shows the mean ratings of the tools on the subscales of the questionnaire on collaborative work. Here, the results

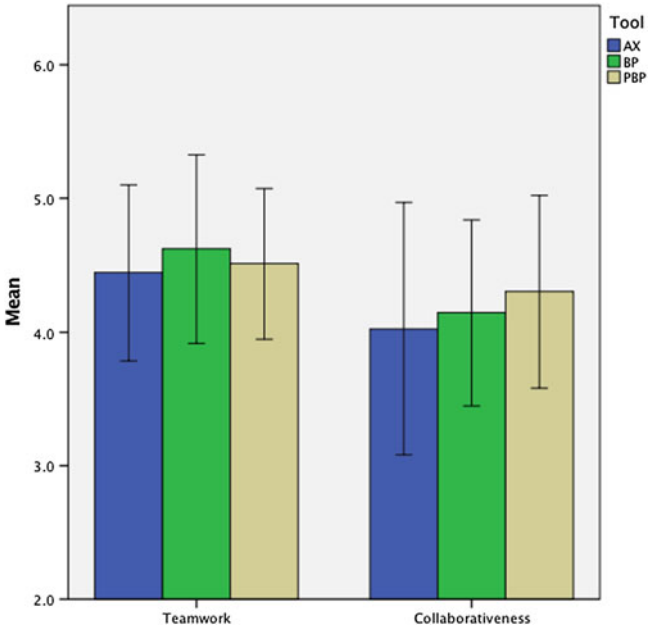


**Fig. 5.4** Boxplots of means for teamwork and collaborativeness. (Error-bars ± SD) (scale ranges from 1 to 7)

draw an inconclusive picture. Again, Axure tends to be rated worse than the other tools. However, the differences between mean ratings are very small. The values vary closely around the middle rating of 4. This tendency to the middle indicates that the used scales were not able to produce a distinctive rating by the test subjects.

As the small differences between the measured means for each subscale, it is no surprise that a repeated measures ANOVA with the factor *tool* did not reveal a statistical trend or significance between the measured means of any subscale.

***Rating of Prototype Results by External Experts***

The results of the mean expert ratings on the prototype results are displayed above in the bar charts in Fig. 5.5. For each subscale, a clear gap can be recognized between the low rated Axure and the both higher rated Blended Prototyping and Paper-based Prototyping.

The differences between the ratings of prototypes created with Blended Prototyping, respectively PBP are comparably small. Prototypes developed in PBP are rated slightly better than those developed in Blended Prototyping in the subscales *look* ($\Delta = 0.391$), *inventiveness* ($\Delta = 0.317$), and *level of maturity* ($\Delta = 0.450$). For the subscale *insights generated* the means for Blended Prototyping and PBP are alike ($\Delta = 0.042$). For the *worth of further investment* ($\Delta = 0.742$) the results created with the Blended Prototyping approach are rated moderately higher than for the PBP approach.

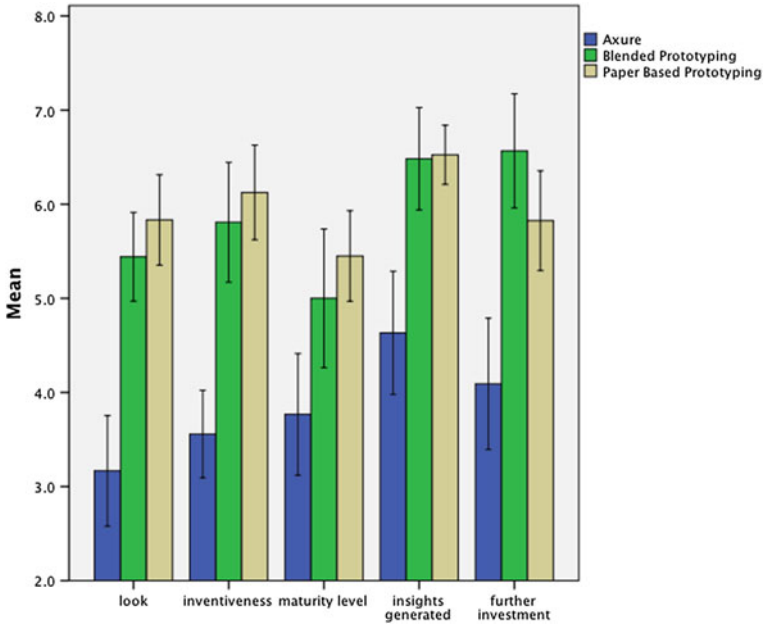


**Fig. 5.5** Boxplots of means for expert ratings of the prototype results; with subscales: *look*, *inventiveness*, *level of maturity*, *insights generated for the further development*, and its *worthiness for a further investment.* Scale ranges from 1 (*very bad*) to 10 (*very good*); (Error-bars ± SD)

To investigate the significance of the measured means for these expert ratings, data was analyzed using a one-way analysis of variances (ANOVA) with the factor *tool* having three levels (Axure, Blended Prototyping, and PBP).

For the subscale *look* ($F\ (2,33) = 7.793$, $p = 0.002$), a Tukey post hoc test revealed significant differences between the measured means between Axure and both other tools (AX and BP: $p = 0.010$; AX and PBP: $p = 0.002$). The difference of means between Blended Prototyping (BP) and PBP bears no statistical significance or trend ($p = 0.854$). Similar statistical significance between the measured means for the tools were revealed in the ANOVAs for the subscales *inventiveness* ($F\ (2.33) = 6.726;\ p = 0.004$) and *generated insights* ($F\ (2,33) = 4.250;\ p = 0.023$): the measured mean for Axure is significantly lower than those of the other two tools in their *inventiveness* (*AX and BP:* $p = 0.012$; *AX and PBP:* $p = 0.005$) and their *generated insights* (*AX and BP:* $p = 0.045$; *AX and PBP:* $p = 0.040$).

For the measured means of the subscale *maturity* ($F\ (2,33) = 1.911;\ p = 0.164$) no statistical significances or trends were revealed.

For the subscale *worthiness of a further investment* ($F\ (2,33) = 4.258;\ p = 0.023$) the ANOVA revealed statistically significant means between the tools Axure and Blended Prototyping ($p = 0.020$). ANOVAs comparing the means

between Axure and PBP *(p = 0.130)* and Blended Prototyping and PBP *(p = 0.674)* were not able to show significant contrasts.

***Ratings of the Collaborative Behavior Analysis from Video Reviews***

The rating with the adapted OTAS schema was conducted for all prototyping assignments. This judgment of the collaborative behavior during the tool-use was very time extensive. For this reason, and for a better objectivity of the raters, three student workers were employed to conduct the video analysis.

All together, 12 * 3 = 36 assignments were created in the course of the user test. The processing time for each of these assignments was limited to 90 min. Hence, a total of 36 * 90 min = 3240 min = 54 h of video sources had to be surveyed. The mean time effort for the judging of video was 110 min, therefore the workload for a post analysis, where each video was just reviewed once, was already 66 h. For the reason of this high effort, reviewing the videos more than once was not affordable to the research project. Three students were involved in the ratings, which each analyzed 4 group-walkthroughs, which is equal to an amount of 12 prototyping sessions.

The mean ratings of the sessions for each subscale of the OTAS rating schema, separated for each tool condition, are displayed in the bar chart in Fig. 5.6. Most subscales show PBP to be rated best, followed by Blended Prototyping and then Axure. Two exceptions can be found: for the *Awareness & Monitoring*, where Axure is rated higher than Blended Prototyping, and for the *Feedback Behavior* where scores for Blended Prototyping and Axure are approximately the same. The measured means for each subscale are relatively close to each other. A repeated measure ANOVA with the factor tool was not able to show any statistical significance or trends for the differences in the measured means for any of the subscales.

As explained above, the correct application of the OTAS schema relies on the proper training of the employed raters. Before the actual rating sessions, the raters were carefully introduced into the meaning and application of the OTAS schema and comparative pre-tests showed well comparable results. However, the actual rating took a very long time, tiredness and annoyance might have affected the coders' ratings in the course of their work. Therefore further analysis was conducted, were the reliability of the subscales was tested. For this, each of the coders was asked to rate two more videos; one selected from previous ratings of each of the other coders. The comparison of the ratings therefore included 6 data pairs.

Table 5.3 shows the results of the conducted inter rater agreement analysis conducted with the intra-class correlation coefficient ICC 1 by Landis and Koch [90]. As a rule of thumb, inter coder agreement of scales investigated with the ICC 1 should be considered to be *poor* for an ICC score smaller than 0.2, *fair* for an ICC score between 0.2 and 0.4, *moderate* for an ICC score between 0.4 and 0.6, *strong* for an ICC score between 0.6 and 0.8, and *almost perfect* for an ICC score above 0.8.

It can therefore be stated, that the sub scales *backup behavior (0.183)* and *team leadership (0.226)* yielded an ICC 1 score that documents less than moderate

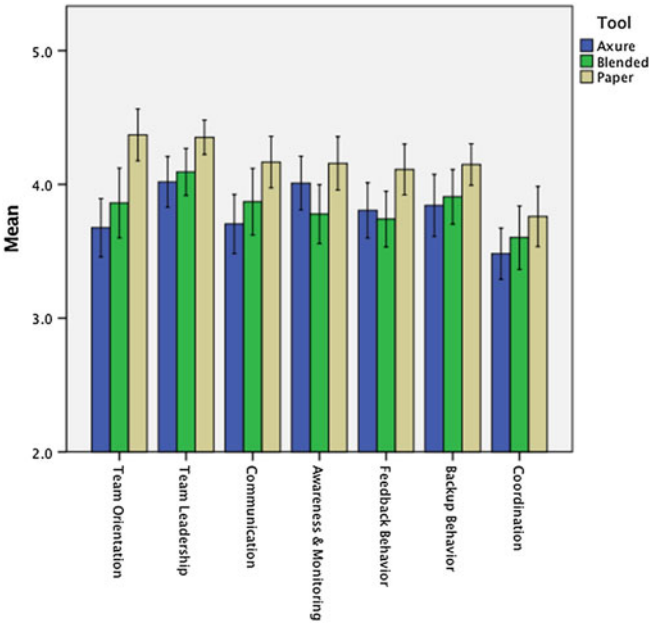


**Fig. 5.6** Boxplots of means for ratings of group-work behavior in the design sessions; with subscales: *team orientation*, *team leadership*, *communication*, *awareness & monitoring*, *feedback behavior*, *backup behavior*, and *Coordination;* Scales range from 0 (*very bad*) to 6 (*very good*); (Error-bars ± SD)

**Table 5.3** Results of investigating the single OTAS scales with ICC 1(ICC 1 scores in normal writing = *poor or fair*, in italic and bold = *moderate*, and in bold = *strong* inter coder agreemente)

| | AX_mean | BP_mean | PB_mean | Sig. | ICC 1 |
|---|---|---|---|---|---|
| Team orientation | 3.569 | 3.819 | 4.347 | 0.139 | **0.738** |
| Team leadership | 4.125 | 4.097 | 4.472 | 0.255 | 0.226 |
| Communication | 3.597 | 3.903 | 4.236 | 0.266 | ***0.433*** |
| Awareness & monitoring | 4.153 | 3.792 | 4.292 | 0.35 | **0.631** |
| Feedback | 3.792 | 3.750 | 4.069 | 0.541 | ***0.467*** |
| Backup behavior | 3.944 | 3.917 | 4.333 | 0.334 | 0.183 |
| Coordination | 3.542 | 3.736 | 3.764 | 0.772 | **0.547** |

agreement between the raters. It is therefore likely, that the coders did not shared a common understanding on how to evaluate a team-process according to the *Backup-Behavior* and *Team Leadership*. Consequently, ratings done on these scales in the post analysis of the recorded design sessions should not be regarded in interpreting the tools' ability to facilitate collaborative work. For the other OTAS sub scales, an at least moderate agreement is shown in the inter rater agreement analysis.

PBP yielded slightly better results than the other two tools in each of the investigated requirements. Between Blended Prototyping and Axure the contrast is even less clear. Here, Blended Prototyping scored better for the *team orientation*, *communication*, and *coordination*. Axure was better rated for *awareness & monitoring*. The results for the measured *feedback behavior* were very much the same between Blended Prototyping and Axure.

In summary, no statistically significant results for the ratings of the collaborative work on the basis of the OTAS rating scheme were found. A trend can be identified that PBP is rated best for all sub scales, Blended Prototyping is slightly better than Axure for all scales but the *Awareness & Monitoring*, where Axure was rated slightly better than the Blended Prototyping approach.

***Rating of the Novelty of Delivered Prototype Results***

To measure the novelty index according to Shah and Smith [133], in the first step a catalog of feature-requirements has to be created for each task. In these catalogs, task specific functional requirements are collected that solutions are expected to address (e.g.: Feature for an index-card learning app (T3): Create index-cards). In the second step, all different implementations for the feature requirement are listed (e.g.: create card from photo, create card by typing, etc.). On this basis, the novelty score is calculated for each solution, depending on the regarded implemented features and their scarcity.

The created catalogs are attached to this work in the appendices X to Y. In summary, for the first task 10, the second task 14, and the third task 7 functional requirements were formulated, each addressed to on average 3 different forms of implementation. The number of functional requirements identified for the single tasks does not speak for its overall creative complexity. The complexity to find solutions for the single functional requirements cannot be directly compared between the tasks.

The novelty score of Shah and Smith allows a weighing of the categories in accordance to their importance to the task solution. The novelty analysis for this user-test was conducted without such weights, since it was hard to find a clearly justified order for the relevance of the identified categories. Hence, all categories for each task were weighted with equal shares.

The means of the calculated novelty scores for the prototypes created with the three different tools are displayed above in Fig. 5.7. Differences between the means are low. Slightly best novelty results were yielded with the Blended Prototyping tool (mean = 0.361), followed by the PBP tool (mean = 0.303), and finally the Axure software (mean = 0.286). Consequently, a conducted one-way ANOVA with the factor *tool* ($F(2, 33) = 1.044$; $p = 0.364$) was not able to show statistical trends or significances between the measured means.

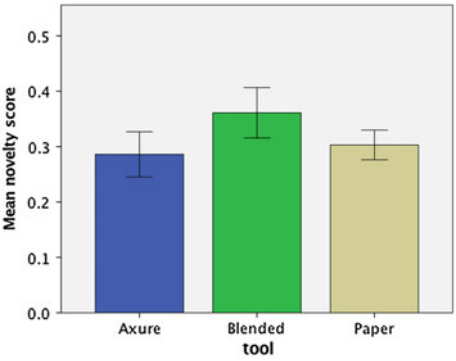


**Fig. 5.7** Boxplots of means for novelty score, judged on the basis of created prototypes. (Values vary from 0 *low* to 1 *high*; Error-bars ± SD)

### (3) Implications of the Measured Data for the Identified Performance Indices

#### *Time-Efficiency of the Prototyping*

As discussed above, to include the ideation processes into the consideration, the ability of a tool to deliver prototypes in a time-efficient manner should be assessed on the basis of the quality of the prototype output, created in a fixed time. Therefore, the measured ratings of external experts are key to the investigation of this performance index.

These expert ratings show weaker results for Axure in comparison to the two other approaches for each of the rated subscales. As displayed in detail above, this weakness was proven to be statistically significant for nearly all of the scales. The results therefore show clearly that the Axure tool is in comparison not well suited to produce prototypes in a time-efficient way.

Comparing the Blended Prototyping and PBP tools shows a less clear picture. Where the paper-based approach shows slightly better ratings for the *look*, *inventiveness*, and *maturity level* of the delivered solutions, the Blended Prototyping approach shows a slight strength towards PBP in producing prototypes the experts would likely follow with *further investments*. None of the described differences between PBP and Blended Prototyping showed to be statistically significant.

Which of the sub scales should have a higher impact in the consideration of the overall prototype quality depends on the specific information goals that are targeted with the prototype. It may therefore be concluded, that Blended Prototyping has an advantage over PBP when the design team primarily wants to assess the potential of an idea to prevail in the development, whereas the PBP approach is better suitable for evaluating factors like the prototype design or its inventiveness. However, the results for Blended Prototyping and PBP in the expert ratings are closely rated.

#### *Collocated Group Work*

No clear conclusions can be drawn from the measured results regarding the performed group work. For the measured self reported group work performance with

the questionnaire by Sauppé and Mutlu [129], no clear differences were shown for any of the investigated sub scales.

For the analysis of the group work performance, based on the examination of the recorded prototyping sessions, no significant differences were measured either. An analysis of the inter-rater agreement in the application of OTAS rating scheme revealed a very weak consistency in the raters' judgment for the sub scales *team leadership* and *backup behavior*. Consequently these scales cannot be regarded in the further analysis.

For the remaining five OTAS sub scales, a trend was identified that PBP is generally rated higher than the other two tools. Blended Prototyping seems to be slightly better rated than Axure for the scales *team orientation, communication,* and *coordination*, whereas Axure was rated slightly higher than Blended Prototyping for the scale *Awareness & Monitoring*.

***Freedom of Creativity and Design***

Regarding the assessment of the ability of the tools to support the freedom of creativity and design in the prototyping process, two sources of measurements were used: ratings by external experts and the application of a novelty score suggested by Shah et al. [133].

The most relevant scores from the expert rating for the *freedom of creativity and design* are the *look* of the prototype, its *inventiveness* and implicitly its *generated insights* and qualification for *further investments*. For all these scales, prototypes designed with Blended Prototyping and PBP scored significantly higher than those that were created with Axure. The ratings for PBP were slightly better than those of Blended Prototyping for the dimensions *look* of the prototype and its *inventiveness*. For the ratings regarding the categories that implicitly indicate the freedom of creativity and design, Blended Prototyping scored slightly better for *further investment* and both tools scored approximately equally for *generated insights*.

The differences in the measured novelty score were not statistically significant. However, its results draw a similar picture of the freedom of creativity and design like the expert ratings. Here prototypes generated with Blended Prototyping are slightly higher rated than those with PBP, Axure is a yet again delivering the weakest results.

In summary, the results show clearly, that the support of freedom and creativity with Axure is weaker than this of PBP and Blended Prototyping. Between PBP and Blended Prototyping the difference is less clear. However, the slightly better results in the most important scales rated by experts gives weak indications that PBP might be better suited to fulfill the *freedom of creativity and design* requirement than Blended Prototyping.

***Usability and User-Experience***

Data related to the usability and user-experience of the surveyed prototyping tools was measured with the AttrakDiff questionnaire. As displayed by the author of the questionnaire Marc Hassenzahl and Sarah Diefenbach [128], the scale *pragmatic*

*quality* can be used as an indicator for the usability of the surveyed tool. Furthermore, the user-experience is related to the two sub scales measuring the *hedonic quality* of the tool.

As pointed out above, the scores for the *pragmatic quality* of Blended Prototyping and PBP were approximately the same. However, both tools scored significantly higher in the scale than the Axure software. Therefore, regarding the tools' usability, advantages of Blended Prototyping and PBP towards Axure can be documented.

Regarding the user-experience, the sub two scales *hedonic quality-identification (HQ-I) and hedonic quality-stimulation (HQ-S)* are to be investigated. For the sub scale *HQ-I* the Blended Prototyping scored slightly higher than PBP, which again scored slightly higher than Axure. Statistical significances could not be shown between either of the measured values, however, a statistical trend between Blended Prototyping and Axure was found. For the sub scale of HQ-S, Blended Prototyping scored significantly higher than both other tools, wherefore it is perceived to provide novel and interesting functionality or interaction styles that stimulate its users in pursuing their personal development goals [128]. Regarding the user-experience of the tools it can therefore be stated, that Blended Prototyping is better able to provide a positive user-experience, especially with regard to the users' hedonic stimulation.

## 5.4 Conclusions of the Comparative Evaluation

The generated results of comparative evaluation show disadvantages of the Axure approach towards the PBP and Blended Prototyping tools in all surveyed requirements. The PBP approach is usually scored slightly better than the Blended Prototyping approach, except for a limited number of subscales of the used assessment methods.

The comparative evaluation measured the application of the compared tools in productive group work sessions in early ideation stages. Consequently the results only speak for such early phases, and no direct conclusions can be drawn for later design stages. However, as highlighted in the discussion of some of the not further investigated requirements above in Sect. 5.2, the software Axure will most probably play out its advantages when investigating later development stages, where a test in mobile context with a higher number of users becomes more relevant.

The Blended Prototyping approach is conceived as transitional approach, which bridges the gap between available tools for earliest design stages and those that are applied later in the development. The PBP approach is perfectly applicable in earliest group work design stages [140], which is underlined by the results of the comparative evaluation as well. Therefore, it is one of the major goals of Blended Prototyping to adapt the advantages of the paper-based approach.

Consequently, it can be seen as a big success that Blended Prototyping performed so similar to the PBP approach.

Significant advantages of Blended Prototyping towards PBP were measured in the AttrakDiff subscale hedonic quality-stimulation. This subscale involves factors that refer to the innovativeness of the tool; therefore it is not astonishing that the high tech tabletop computing based Blended Prototyping scores higher in this dimension. Furthermore, Blended Prototyping scored significantly higher in the expert ratings for the worth of a further investment of the yielded prototype results. This is an interesting observation which underlines, that the questioned domain experts consider prototypes that are able to be ran on a mobile device as to be more valuable to follow than those which are still in a paper state.

An observation that appears to be confusing on first sight can be made looking at the OTAS team result data regarding the *Awareness & Monitoring*, where Axure scored higher than Blended Prototyping. The data is not significant, however the inter coder reliability for this scale was comparatively high. When the user tests were conducted, the low-tech design tool methods of Blended Prototyping were at a state not reliable enough to be used in a productive work scenario. Therefore in the test, the users had to stick to use a tablet device to control the design tool environment. This was likely affecting the awareness & monitoring within the team since only one user can use the tablet at a time.

Further comparative tests are necessary that investigate the work with the tools on later design stages. Here, other requirements become more relevant, wherefore the performance of the Axure software will likely be better than observed in test results discussed above. Such tests should necessarily involve real user tests in the field. It will be interesting to see, how the Blended Prototyping approach will perform at later stages in comparison to Axure. As discussed above in Sect. 4.1 Blended Prototyping addresses a number of challenges to allow for a mixed-fidelity prototyping in the actual use context.

# Chapter 6
# Conclusion and Future Work

This concluding chapter summarizes the results of this work and puts them in the context of the research questions, postulated for this work. It points out the contributions of this work for research and practice, and discusses next steps of a further evaluation of the developed processes and tools in Blended Prototyping, regarding its role in later development stages.

Furthermore, the chapter reflects future work that should be addressed to the topic, which enhances the Blended Prototyping platform in its current state, and reflects other possible application domains.

## 6.1 Conclusion

The first research question **(RQ 1)** on how to develop a catalog of requirements for prototyping systems for the development of mobile app user interfaces was addressed in Chap. 3. In a first step to develop such a catalog, requirements discussed in related work were collected in a literature review that systematically analyzed related work in the most relevant conferences and journals to the topic. This first collection was then used in an expert study, where practitioners with a profound experience in the design and development of mobile apps rated the importance of the suggested categories for five different design stages. Moreover, the experts were asked to name additional requirements that could contribute to the collection yielded from literature.

As a result from this expert evaluation, lists about the most important requirements for five different project stages of mobile app development projects were generated, ranging from *very early* to *very late* stages. The expert ratings lead to a rejection of one of the requirements identified from literature, the *automated model based evaluations* that was rated low for each development stage. The resulting catalog therefore includes 16 requirements that were derived from literature and rated to be important tool requirements. In addition to this, the catalog covers



B. Bähr, *Prototyping of User Interfaces for Mobile Applications*,
T-Labs Series in Telecommunication Services,
DOI 10.1007/978-3-319-53210-3_6

additional requirements that were newly suggested by the experts. However, ratings about the five new dimensions were just given by the experts who suggested them and were not further evaluated with others. In summary, the catalog consists of 16 requirements that were evaluated with experts, and an appendix of 5 additional dimensions that display the opinion of single contributors.

The catalog highlights a number of requirements that change in their relevance throughout the different design and development stages. As a consequence, the top ranked requirements vary from one stage to the other.

The catalog may not include totally unexpected categories, however, it provides a systematic view on the requirements of prototyping tools that did not exist yet. From my point of view, too often such requirements are handled in a matter of fact way. Most authors in related work that discus the development of new prototyping tools motivate their work with a limited number of factors, which are more or less taken as granted. However, basing the selection of requirements on personal experiences bares the risk to in fact miss certain factors a system development could benefit from. Here, the prototyping requirements catalog can provide a good orientation point, to gain an idea of aspects that are worth a consideration, with respect to the design-stages the tool aims to support. As such a landmark, the catalog was very helpful in the development of the Blended Prototyping approach.

Regarding the second research question **(RQ 2)**, about the applicability of the requirements catalog in evaluations, application studies were conducted for both, reviews with experts and in a comparative performance analysis.

As described at the end of Chap. 3, experts used the catalog to rate the concept of the Blended Prototyping approach. For this, a short description of the single requirements was sufficient to put the experts into the position to rate the prototyping approach on a 10-point likert scale ranging from '*the requirement is not met at all*' to '*the requirement is fully met*'. In this proceeding, none of the experts had further questions about the requirements, or faced difficulties in giving the ratings.

In a further application, the requirements catalog served as a basis to derive metrics for measuring the performance of different tools in their practical application (compare Sect. 5.2). Here, the lists of requirements for early to middle design stages provided information of the most relevant characteristics for the evaluation, on which basis metrics were derived to measure the tool performance.

The choice of such metrics has to be made under careful consideration of the specific test design. As discussed in detail below, not all metrics chosen for the comparative test described in Chap. 5 delivered meaningful contrasts between the investigated tools. However, this might question the employed metrics, but not the performance categories derived from the requirements catalog.

The answer to the third research question **(RQ 3)**, on how to develop a new mixed-fidelity prototyping approach targeted at a whole list of design objectives, is discussed in detail in Chap. 4, where conception, design, and implementation of the Blended Prototyping platform is discussed. An evaluation of the concept was done with expert ratings, where the requirements catalog developed in the work served as a set of criteria for the judgments.

The approach adapts the paper-based prototyping techniques in its design process to the largest extent. The advantages of designing ideas on the basis of hand sketches on physical paper are facilitated. As displayed in detail in Sect. 2.4.3, pen-and-paper sketching on physical paper can help to a faster and more direct articulation of ideas, which has positive effects on the creative process and on teamwork.

In Blended Prototyping, these hand-sketches on paper are done in a tabletop-computing environment. As displayed in Sect. 2.4.2.3, different related work praises such interactive surfaces for their abilities to embed physical objects into the human-computer interaction in a natural manner, to promote collaborative work in creating a situation where the whole team is involved in the system interaction, and to leverage creative work. The Blended Prototyping tool is built as unobtrusive as possible, to focus in its ideation process with paper sketches rather on the human-human than on the human-computer interaction. The system allows for a quick generation of mixed-fidelity click-dummy prototypes, and at the same time offers processes to program advanced prototype functionality in native code. The approach therefore blends the paradigms of quick throwaway prototyping and sustained evolutionary prototyping. Blended Prototyping supplies tools for distributing, testing, and analyzing the developed prototypes directly on the target devices, wherefore specific usability challenges for the prototype, that derive from mobile use contexts, can be investigated in user studies.

In a rating of the concept of Blended Prototyping with experts, where the catalogue discussed above was applied, the approach was judged generally positively. The system is designed to provide a helpful prototyping tool primarily for early to middle design phases of mobile apps. Especially for the most important requirements to these stages, Blended Prototyping was judged with high ratings. Considering the five most important dimensions for early to middle stages, on a 10-point likert scale ranging from 1 to 10, the approach scored 9.1 for *getting quick prototypes*, 9.1 for *collocated group work*, 7.1 for *freedom of creativity and design*, 7.6 for *simultaneous design of different ideas*, and 7.5 for *support expert reviews.*

Regarding the fourth research question **(RQ 4)**, on whether the Blended Prototyping approach is able to improve the prototyping process of mobile app user interfaces in accordance with the identified requirements, an application study comparing three prototyping tools has been conducted, as described in Chap. 5. Here, the Blended Prototyping approach was evaluated in direct comparison to the Paper-Based Prototyping (PBP) approach and the software tool Axure.

As displayed in detail in the introduction to the PBP approach in Sect. 2.3, the technique is very well suited for the prototyping in early design stages. The limitations of the process to conduct tests in mobile use contexts, which is advisable for testing mobile app user-interfaces, was not regarded in the study in Chap. 5, since here the prototype tests took place in a stationary setting. As a consequence, it was expected that the PBP approach would yield the best results in the experiment.

In contrast to that, the use of Axure in the given study design was likely to encounter a number of limitations. First, using software that is designed for a professional application requires a certain level of expertise. Since the study

participants in general did not have previous knowledge about using Axure, negative effects on the prototyping process and its results were expected. However, the participants received an introductory training where the most important software functions in order to create click-dummies in Axure were explained. More advanced functionality that requires more in-depth mechanisms of the Axure software, were not used in any of the created prototypes.

Second, Axure is a software solution that is used at a stationary computer, from which problems arise in the collaborative work. The fact, that in group work that shares a stationary computer not all team members have the same access to the system control and data, can result in lower participation of those team members who feel excluded from the design process. This being said, the effect grows with group size and might be comparably small for groups of three members.

Not all of the applied metrics were equally successful in identifying differences between the compared prototyping processes. Doubts have to be articulated towards the ability of the applied methods to investigate the tools' ability to facilitate ***collocated group work***. The employed questionnaire produced very similar results for all tested tools. That was surprising, given the fact that related work attributes PBP with an especially high capability, and Axure with an especially low capability to facilitate collaborative work.

The questionnaire regarding collaborative work did contain questions on the perception of the participants' interpersonal relations as well. These were important to consider, since particularly bad interpersonal relations in a group can easily jeopardize the whole group work process and outcome, regardless of the applied tools. However, the participants therefore had to fill out a very lengthy questionnaire after each test session, where they were asked about their interpersonal relations to each of the two other team members. Many situations were witnessed, where participants reacted annoyed about filling out these lengthy questionnaires. This annoyance might have affected the participants' ratings of the questions regarding collaborative work in that questionnaire as well.

The conducted post-analysis, to gain a judgment of the collaborative work on basis of the videotaped prototyping processes, was not able to produce significant results either. Investigating the inter-coder reliability of the ratings showed that the raters did not have a shared agreement on rating two of the seven rating categories. Here, a better training of the raters into the rating scheme OTAS might have been necessary, or the amount of work for the coders might have been too high. For the remaining categories on collaborative work slightly better results in the tool performance were identified for PBP. However, the contrasts were much too small to be able to make a meaningful statistical statement.

The other investigated requirements produced more meaningful results. For the tools' ***time-efficiency*** the study showed statically significant lower results for Axure in comparison to Blended Prototyping or PBP. The differences in the mean ratings related to time-efficiency regarding PBP and Blended Prototyping were small.

Investigating the requirement ***freedom of creativity and design*** draws a similar picture. Again, the results of Axure were statically significant lower than these of the other two, whereas the ratings for Blended Prototyping and PBP were similar.

Here, PBP scored slightly better in expert ratings of the *look* and *inventiveness* of the prototype result, whereas its *generated insights* and probability for a *further investment* were rated slightly better for Blended Prototyping.

Applying the AttracDiff scale *pragmatic quality* as an indicator for tools ***usability***, both PBP and Blended Prototyping scored equally high and both statically significant better than Axure. The scales *hedonic quality-stimulation (HQ-S)* and *hedonic quality-identification (HQ-I)* were used to investigate the ability of the tool to facilitate a positive ***user experience***. For *HQ-I* the Blended Prototyping scored slightly higher than PBP, which again scored slightly higher than Axure. Statistical significances could not be shown between either of the measured values, however, a statistical trend between Blended Prototyping and Axure was found. For the sub scale of HQ-S, Blended Prototyping scored significantly higher than both other tools, which means that Blended Prototyping is perceived to provide functionality or interaction styles of higher novelty or interest.

In summary, the comparative user-test therefore shows that PBP and Blended Prototyping are better suited to be applied for group work of early design stages. The results support the assumption that Axure performs comparably weak in the tested conditions. The differences between Blended Prototyping and PBP are small for each recorded measurement. This shows the success of Blended Prototyping to in fact adapt the advantages of the PBP approach!

The results of the comparative application study described above evaluate the tools is specific conditions that did not allow the investigation of all relevant requirements dimensions. The applicable of a tool to be used in *design reviews* or *expert reviews* are important requirements that were not tested in the experiment. These should be investigated in further studies that are targeted at exactly these factors.

Moreover, the ability of the prototypes to *support tests in the real use contexts* was not investigated in the study, since the prototype tests were held in a stationary context. Investigating this requirement will most probable lead to results where the PBP approach reveals weaknesses. Here, the capabilities of Blended Prototyping and Axure are interesting to compare.

Another key advantage of Blended Prototyping did not find consideration in the test either: the programming with *reusable code*. Here, comparative evaluations with the ordinary standard development tools and Axure should investigate, whether the processes in Blended Prototyping are in fact able to simplify the programming of more advanced prototype functionality.

Moreover, it will be interesting to investigate a real-life application of Blended Prototyping, where the platform is actively used for a longer period of time in the daily business of a design agency.

An important aspect of the Blended Prototyping approach is to support of trans-disciplinary groups. In the user-study described above, participants were recruited on the basis of their performance in pre-tests regarding their creative capabilities in programming skills. However, most of the recruited participants were still involved in their university studies and their practical expertise was limited. Therefore, it will be interesting to investigate, whether the application of Blended

Prototyping with full-expert users will be able to deliver comparably positive results in facilitating a collaborative and creative work process.

In summary, this work builds two main contributions that are valuable for research, as well as the development of mobile apps in practice. First, it presents a successfully evaluated catalogue of requirements for mobile app prototyping tools that are ranked in their relevance at different development stages. This catalogue provides a good orientation point for the conception of new prototyping approaches and identifies metrics for a comparative performance evaluation of alternative techniques.

Second, it presents the new prototyping technique Blended Prototyping, which was successfully evaluated in a user-test where its performance was compared with the PBP approach and the software tool Axure. The evaluation showed, that in early design stages Blended-Prototyping performs generally better than the software tool Axure. Moreover the approach is able to adapt the advantages of the Paper-Based-Prototyping concepts, which is shown in the very similar test performances of the two techniques.

## 6.2 Future Work

Multiple ideas exist, on how to extend the Blended Prototyping system to improve the prototyping process of mobile app user interfaces in additional aspects.

The requirements catalogue identified in this work includes factors that are not yet supported in the approach. The *cross platform support* of the prototypes could be established by mapping data created in the design sessions not to Java-classes, but to classes of a web technology like Javascript, which is supported by all mobile platforms. However, the use of web-technologies has two downsides. First, it requires the re-programming of the prototype code in the development of the final product. Second, web-technologies do not offer the same capabilities and programming libraries as native technologies.

Moreover, Blended Prototyping does not support *remote collaborative work*. The infrastructure of a full-duplex synchronization of the systems data via web-sockets is already established, since the technology is currently used for the communication between the design tool computer and the mobile control clients. This technology could be extended to produce a setup, where multiple design tables are used in a creative process at different locations. Here, different related work exists [25, 27, 121] that addresses the issue of handling and organizing tangible objects in a collaborative remote work context.

Another yet unfulfilled requirement is the *use of animations*. Animations can play a key role in the design of mobile user interfaces, not just in the development of mobile games. However, the definition of animations requires extensive programming that is hard to realize in a rapid prototyping context. Addressing this context, first versions of alternative approaches are already implemented on how to define animations in the described tabletop design tool. Two of these address

animation design with gestures on mobile tablet devices, which are inspired by the new AutoDesk tool Draco [74]. Another approach adapts the ideas of Walther-Franks et al. [156] of a full-embodied definition of animations.

Moreover, other ways should be explored to allow the definition of as much functional complexity as possible in the design tool, without overwhelming the simplicity of the current approach. For this, more user-controls could be included to extent the current definition space. Here, the prototyping tool Balsamiq gives an impressive example by supporting a set of 77 different user controls.

Even more functional aspects could be implemented by extending the Blended Prototyping technology with codeless-programming techniques. This way, easy rule-based behavior could be defined directly in the design tool. Block based visual programming approaches like the MIT-Scratch [96], or its adaption Google Blockly [172], can provide excellent starting points.

The Blended Prototyping platform will be made publicly accessible under an open source license. In the design of the system, clear and open standards were implemented where data is transferred from one module to the other. Therefore, it is easy to substitute single modules with different implementations, and still take advantage of other aspects of the platform. This way, the system development could be proceeded by others, who find valuable applications of the system in research or practice.

As a follow-up research project to the topic I currently plan a similar application for the collaborative interdisciplinary design of Internet of Things (IoT) applications. Such applications usually run as a combination of mobile devices that access distributed sensors. Here, an extension of the tabletop design tool could serve as a good platform to explore the meaning and capabilities of different sensor elements, and combine them in conjunction with a mobile app to new hardware/software prototypes. Software prototypes could be built in a similar fashion as described in this work, hardware prototypes could be built out of programmable Arduino microcontrollers and modularized sensor frameworks like grove sensors[1] or Google Beacons.[2]

As technical systems developed fast in the past, they will develop fast in the future. Though, in a recent Nature article, M. Mitchell Waldrop claims an end of the long time valid Moore's Law approaching [153], mobile and ubiquitous devices are very likely to become more and more part of our daily life in the future. Be it the Internet of Things or the advent of the wearable's, the design of user interfaces, where the complex systems of the human mind is interacting with complex computer systems, will always stay a big challenge.

As a consequence, the prototyping of user-interfaces will be a relevant topic in the future, as it is today. Furthermore, sketching, or similar techniques of expression, which can help designers to seamlessly express themselves, will always be of

[1] http://www.seeedstudio.com/wiki/Grove_System.

[2] https://developers.google.com/beacons/.

interest for the conception of creation and ideation tools. Similar fields of expression might be found in speech, gestures, or even brain-computer interfaces.

Prototyping can teach us to be brave, to communicate what we have in mind, and to open up ideas that might not yet be perfect. Of course, opening up always risks to suggest something stupid; but many who suggested something great, were looked upon first in skepticism and bewilderment.

# Appendices

# Appendix A
# Email-Subscribe Code Example

```
package baehrben.developments.subscribeDialog;

import
baehrben.developments.ttsmippsAndroid.classStructure.IF_ScreenToAndroidBase;
import baehrben.developments.ttsmippsAndroid.classStructure.TTSMIPPS_Screen;
import baehrben.developments.ttsmippsAndroid.classStructure.W_Button;
import baehrben.developments.ttsmippsAndroid.classStructure.W_TextBox;
import android.content.Context;
import android.util.Log;

public class Screen11 extends TTSMIPS_Screen {

   final static String imagePath = "Screen11.png";
   final static String prototypeName = "subscribeDialog";
   W_Button scBg_11_btn_1;
   W_TextBox scBg_11_tb_1;
   IF_ScreenToAndroidBase delegateToBase;
```



B. Bähr, *Prototyping of User Interfaces for Mobile Applications*,
T-Labs Series in Telecommunication Services,
DOI 10.1007/978-3-319-53210-3

```
public void handle_btnScr11_1() {

    // Edit code here (start)
    if (emailIsValid()) {
      Screen14 targetScreen = new Screen14();
      delegateToBase.changeToScreen(targetScreen);
    }
    else {
      Screen13 targetScreen = new Screen13();
      delegateToBase.changeToScreen(targetScreen);
    }

    // Edit code here (end)

  }
}
  // TODO: emailIsValid()
}
```

# Appendix B.1
# Discarded Collaboration Coding Schema

| alle 5 Minuten den Fragebogen ausfüllen: | | Codierung: 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| allgemein | in welcher Form haben die Gruppenmitglieder gearbeitet? | jeder für sich | gemeinsam | | | |
| Communication | Haben sich die Gruppenmitglieder in gleichem Umfang am Gespräch beteiligt? | eine Person dominierte komplett | eine außen vor; andere unausgewogen | eine außen vor; andere ausgewogen | ja, sehr gleiche Beteiligung | |
| | Wie häufig wurden Demonstrativpronomen verwendet? (z.B. das hier, dieser, dort) | gar nicht | 1-2x | 3-5x | >5x | |
| | In welcher Frequenz haben die Gruppenmitglieder Augenkontakt gehabt? (bzw. sich angeschaut) | gar nicht | . | . | . | permanent |
| Awareness (negativ) | Wie häufig haben sich die Gruppenmitlgieder gegenseitig unterbrochen (im Gespräch/ bei der Ausführung einer Aktion) und damit das laufende Gespräch/ die Aktion abgebrochen | gar nicht | 1x | 2-3x | >3x | |
| | Wurden vorher getroffene Absprachen oder Abläufe neu diskutiert oder unabgesprochen verändert? | gar nicht | 1x | 2-3x | >3x | |
| | Haben sich die Gruppenmitglieder bei der Ausführung der Aufgabe unabsichtlich gegenseitig behindert? (z.B. die Sicht blockiert) | gar nicht | 1x | 2-3x | >3x | |
| | Wie oft haben die Gruppenmitglieder sich gegenseitig nach ihrem Vorhaben oder vergangenen Aktivitäten befragt? (z.B. "was tust du gerade?", "wofür ist das?", "warum hast du das so gemacht?") | gar nicht | 1x | 2-3x | >3x | |
| | wurden Informationen wiederholt besprochen/ nachgefragt? | gar nicht | 1x | 2-3x | >3x | |
| Awareness (positiv) | Haben die Gruppenmitglieder in konstruktiver Weise parallel an einer (Teil-) Aufgabe gearbeitet, ohne sich vorher abzusprechen? | nein | ja | | | |
| | Haben sich die Gruppenmitglieder ohne Absprache Objekte herübergereicht (z.B. Stifte, Papier) | nein | ja | | | |
| | Haben sich die Gruppenmitglieder bei ihrer Tätigkeit gegenseitig ohne Absprache ergänzt (z.B einer bedient die Maus, der andere die Tastatur; einer hält das Papier fest, der andere legt an) | nein | ja | | | |
| Awarness work | Wurden eigene Aktionen laut kommentiert, um den Gruppenmitgliedern das eigene Vorgehen deutlich zu machen? | nein | . | . | . | permanent |
| Coordination | wurden Aktionen ungewollt von verschiedenen Gruppenmitgliedern doppelt ausgeführt? | nein | ja | | | |
| | Wurde die Bearbeitung der Aufgabe aufgehalten, dadurch, dass gleiche Ressourcen parallel genutzt wurden? | nein | ja | | | |
| | Sind die Gruppenmitglieder während der Bearbeitung unabsichtlich gegeneinander (oder beinahe gegeneinander) gestoßen? (z.B. beim Greifen nach Material; Verändern der eigenen Position) | gar nicht | 1x | 2-3x | >3x | |
| | Waren alle Gruppenmitglieder gleichmäßig in die Zusammenarbeit eingebunden? | nein | ja | | | |
| Besonderheiten / Kommentar | | | | | | |

# Appendix B.2
# Applied OTAS-Coding Schema

| Kategorie | Erklärung |
|---|---|
| Team orientation | Die Einstellung der Teammitglieder gegenüber einander und der gemeinsamen Aufgabe. Bereitschaft sich zu engagieren und die Teamziele ernst zu nehmen |
| Team leadership | Eine oder unterschiedliche Personen achten darauf, dass die Bearbeitung strukturiert abläuft und zum Ziel führt |
| Communication | Kommunikation untereinander (verbal, durch Körpersprache) zur erfolgreichen Bearbeitung der Aufgabe |
| Awareness and monitoring | Die Teammitglieder wissen jeweils, was die anderen gerade machen bzw. vorhaben |
| Feedback | Feedback zur Bearbeitung der Aufgabe wird gegeben oder erfragt. Feedback wird angenommen |
| Backup behavior | Die Teammitglieder unterstützen sich gegenseitig, wenn der andere gerade nicht weiter weiß oder Fehler macht |
| Coordination | Die Arbeit verläuft zeitlich sinnvoll und geht Hand in Hand<br>Es entstehen keine Cooperation-Breakdowns (Zugriff auf gleiche Ressourcen, Behinderung der Arbeit der Gruppenmitglieder) |

# Appendix B.3.1
# Pre-questionnaire—General Questions

Wie alt sind Sie?

_____

Von welchem Geschlecht sind Sie?

☐ weiblich
☐ männlich

Wie bezeichnen Sie Ihren Beruf?

_____

Wie viele Jahre arbeiten Sie bereits in Ihrem Beruf?

_____

Wenn Sie studieren, welches ist Ihr Studienfach?

_____

Wenn Sie studieren, wie viele Jahre studieren Sie?

_____

Nutzen Sie ein Smartphone? Wenn ja, von welchem System?
(mehrfach-Antwort möglich)

☐ ich nutze kein Smartphone
☐ ich nutze ein iPhone oder iPad
☐ ich nutze ein Android Gerät
☐ ich nutze ein Windows Mobile Gerät
☐ ich nutze Blackberry Gerät
☐ anderes: _____

# Appendix B.3.2
# Custom Creativity Assessment Questionnaire

Bitte kreuzen Sie für jeden Bereich die Aussage an, die am ehesten für Sie zutrifft.

*A. Visuelle Kunst (Gemälde, Zeichnungen, Skulpturen, ..)*

☐ Ich habe keine Erfahrungen mit visueller Kunst (weiter mit Musik).
☐ Ich beschäftige mich damit ab und an in meiner Freizeit.
☐ Ich beschäftige mich damit häufig in meiner Freizeit.
☐ Ich habe längere Zeit Unterricht in diesem Bereich bekommen.
☐ Ich habe meine Kunst schon mal ausgestellt/ an Wettbewerben damit teilgenommen.
☐ Ich habe schon Geld mit meiner Kunst verdient.
☐ Ich arbeite in diesem Bereich.
☐ Ich habe schon Auszeichnungen dafür gewonnen.

*B. Musik*

☐ Ich habe keine Erfahrungen mit Musik (weiter mit Tanz).
☐ Ich mache ab und an in meiner Freizeit Musik.
☐ Ich mache häufig in meiner Freizeit Musik.
☐ Ich habe längere Zeit Musikunterricht bekommen (abgesehen vom Musikunterricht in der Schule).
☐ Ich habe schon mal vor Publikum Musik gespielt/ mit meiner Musik an Wettbewerben teilgenommen.
☐ Ich habe schon Geld mit meiner Musik verdient.
☐ Ich arbeite in diesem Bereich.
☐ Ich habe schon Auszeichnungen für meine Musik gewonnen.

*C. Applikations- und Webdesign*

☐ Ich habe keine Erfahrungen in diesem Bereich.
☐ Ich habe in meiner Freizeit schon ab und zu kleine Aufgaben in diesem Bereich bearbeitet.
☐ Ich bearbeite in meiner Freizeit häufig Aufgaben in diesem Bereich.
☐ Ich habe schon öfters Kurse in diesem Bereich besucht/ Bücher dazu gelesen
☐ Ich habe schon für andere Leute Programme geschrieben/ meine Programme veröffentlicht/ an Wettbewerben mit meinen Programmen teilgenommen.
☐ Ich habe schon Geld mit meinen Programmen verdient
☐ Ich arbeite in diesem Bereich
☐ Ich habe schon Auszeichnungen für meine Programme erhalten.

*D. Programmierung*

☐ Ich habe keine Erfahrungen in diesem Bereich.
☐ Ich habe in meiner Freizeit schon ab und zu kleine Programme oder Skripte selbst geschrieben.
☐ Ich schreibe in meiner Freizeit häufig Programme oder Skripte selbst
☐ Ich habe schon öfters Programmierkurse besucht/ Bücher dazu gelesen
☐ Ich habe schon für andere Leute Programme geschrieben/ meine Programme veröffentlicht/ an Wettbewerben mit meinen Programmen teilgenommen.
☐ Ich habe schon Geld mit meinen Programmen verdient
☐ Ich arbeite in diesem Bereich
☐ Ich habe schon Auszeichnungen für meine Programme erhalten.

*E. Tanz*

☐ Ich habe keine Erfahrungen mit Tanz (weiter mit Architektur).
☐ Ich tanze ab und an in meiner Freizeit.
☐ Ich tanze häufig in meiner Freizeit.
☐ Ich habe längere Zeit Tanzunterricht bekommen.
☐ Ich habe schon mal vor Publikum getanzt/ an Tanzwettbewerben teilgenommen.
☐ Ich habe schon Geld mit tanzen verdient.
☐ Ich arbeite in diesem Bereich.
☐ Ich habe schon Auszeichnungen fürs Tanzen gewonnen.

*F. Architektur und Design*

☐ Ich habe keine Erfahrungen in diesem Bereich (weiter mit kreativem Schreiben).
☐ Ich beschäftige mich ab und an in meiner Freizeit damit.
☐ Ich beschäftige mich häufig in meiner Freizeit damit.
☐ Ich habe schon mehrere Architektur- oder Design-Kurse besucht.
☐ Ich habe meine Projekte schon mal vor Publikum ausgestellt/ an Architektur- oder Design-Wettbewerben teilgenommen.
☐ Ich habe schon Geld mit meinen Projekten verdient.
☐ Ich arbeite in diesem Bereich.
☐ Ich habe schon Auszeichnungen für meine Architektur- oder Design-Projekte gewonnen.

*G. Kreatives Schreiben*

☐ Ich habe keine Erfahrungen in diesem Bereich (weiter mit Humor)
☐ Ich schreibe ab und an in meiner Freizeit.
☐ Ich schreibe häufig in meiner Freizeit/ Ich habe schon eigene lange Texte geschrieben.
☐ Ich habe über längere Zeit Schreibunterricht bekommen (abgesehen vom Deutschunterricht in der Schule).
☐ Ich habe meine Werke schon vor Publikum vorgetragen/ veröffentlicht/ an Wettbewerben damit teilgenommen.
☐ Ich habe schon Geld mit meinen Werken verdient.
☐ Ich arbeite in diesem Bereich.
☐ Ich habe schon Auszeichnungen für meine Schreibwerke gewonnen.

*H. Humor*

☐ Ich habe kein Talent in diesem Bereich (weiter mit Erfindungen).
☐ Viele Leute lachen über meine Witze oder Geschichten.
☐ Ich habe mir schon mal Witze oder lustige Geschichten selbst ausgedacht.
☐ Ich habe über längerer Zeit Unterricht in diesem Bereich bekommen.
☐ Ich habe meine Witze schon mal vor Publikum vorgetragen/ in Büchern veröffentlicht / an Wettbewerben damit teilgenommen.
☐ Ich habe schon Geld mit meinem Humor verdient.
☐ Ich arbeite in diesem Bereich.
☐ Ich habe schon Auszeichnungen für meinen Humor gewonnen.

*I. Erfindungen*

☐ Ich habe kein Talent in diesem Bereich (weiter mit Wissenschaftlicher Arbeit).
☐ Mit fällt es leicht neue (unübliche) Verwendungszwecke für Objekte zu finden.
☐ Ich habe schon Skizzen für eigene Erfindungen entworfen/ Konzepte erarbeitet.
☐ Ich habe schon Prototypen für eigenen Erfindungen erstellt/ digitale Zeichnungen angefertigt.
☐ Ich habe schon komplexere Erfindungen kreiert/ mit meinen Erfindungen an Wettbewerben teilgenommen.
☐ Ich habe schon Geld mit meinen Erfindungen verdient.
☐ Ich arbeite in diesem Bereich.
☐ Ich habe schon Auszeichnungen für meine Erfindungen gewonnen.

*J. Wissenschaftliche Arbeit*

☐Ich habe kein Talent in diesem Bereich (weiter mit Theater).
☐Ich denke häufig über die Lösung von alltäglichen Problemen nach
☐Ich denke häufig über die Lösung von wissenschaftlichen Problemen nach
☐Ich habe schon an wissenschaftlichen Projekten mitgearbeitet.
☐Ich habe schon mal eigene wissenschaftliche Projekte durchgeführt/ mit den Projekten an Wettbewerben teilgenommen.
☐Ich habe schon mal Geld/ Stipendien für meine wissenschaftliche Arbeit erhalten.
☐Ich arbeite in diesem Bereich.
☐Ich habe schon Auszeichnungen für meine wissenschaftliche Arbeit gewonnen.

*K. Theater und Film*

☐Ich habe keine Erfahrungen in diesem Bereich (weiter mit )
☐Ich habe schon mal in einem Theaterstück oder Film mitgewirkt.
☐Ich wirke häufig in Theaterstücken oder Filmen mit.
☐Ich habe längere Zeit Schauspielunterricht erhalten.
☐Ich habe schon mal eigene Theaterstücke oder Filme produziert/ an Schauspiel-Wettbewerben teilgenommen.
☐Ich habe schon mal Geld für meine Mitarbeit in Theaterstücken oder Filmen erhalten.
☐Ich arbeite in diesem Bereich.
☐Ich habe schon Auszeichnungen für meine Theater- oder Filmprojekte erhalten.

*L. Kulinarisch Künste*

☐Ich habe kein Talent in diesem Bereich (weiter mit Programmierkenntnissen).
☐Ich koche auch aufwendige Rezepte für mich zu Hause.
☐Ich koche öfters für andere Leute.
☐Ich denke mir häufig eigene Rezepte aus, die ich koche.
☐Ich habe schon öfters Kochkurse besucht/ an Kochwettbewerben teilgenommen.
☐Ich habe schon mal Geld für mein gekochtes Essen erhalten.
☐Ich arbeite in diesem Bereich.
☐Ich habe schon Auszeichnungen für meine Kochkünste erhalten.

# Appendix B.3.3
# Pre-questionnaire—Java-Questions

1) Welche Aussagen trifft auf die Programmiersprache Java zu?
- ☐Java ist eine web-basierte Programmiersprache
- ☐Java ist eine funktionale Programmiersprache
- ☐Java ist eine objektorientierte Programmiersprache
- ☐Java ist eine skriptbasierte Programmiersprache

2) Welche Anweisungen wird zu einer Ausgabe im Terminal führen?
- ☐Terminal.write(“Hallo Welt”);
- ☐System.out.println(“Hallo Welt!”);
- ☐System.display.Text(“Hallo Welt!”);
- ☐out.write(“Hallo Welt!”);

3) Um in Java Methoden global zur Verfügung zu stellen bzw. den Zugriff zu beschränken werden folgende Zugriffsmodifikatoren benutzt:
- ☐+, -
- ☐pub, priv
- ☐allow, block
- ☐public, private

4) Sie möchten gerne das Wort „Hallo“ in einer Variablen speichern. Welchen Dateityp wählen Sie hierfür am Besten?
- ☐Integer
- ☐String
- ☐float
- ☐boolean

5) Wie viele Klassen können in einer Java-Datei deklariert werden?
- ☐nur eine, die Hauptklasse der Datei
- ☐zwei, die Konstruktorklasse und die Hauptklasse der Datei
- ☐die Anzahl ist nicht beschränkt
- ☐das hängt vom Grad der Vererbung ab

6) In welcher Auswahl ist ein Wort enthalten, welches kein sogenanntes Keyword in Java ist?
- ☐do, while, for
- ☐protected, static, super
- ☐int, always, char
- ☐alle in a, b und c enthaltenen Wörtern sind keywords in Java

7) Zeilen-Kommentare werden in Java wie folgt gekennzeichnet:
- ☐ <!—- —>
- ☐ **
- ☐ #
- ☐ //

8) Wie sollten, laut Namenskonvention, alle Klassen-Namen in Java anfangen
- ☐ mit einem .
- ☐ mit einem großen Buchstaben
- ☐ mit einem _
- ☐ mit einem ::

9) Welche Schleifenschreibweise gibt es nicht
- ☐ for(int i=0;i<8;i++){}
- ☐ int i=0; until (i == 8){i ++;}
- ☐ int i=0; while(i<8){i++;}
- ☐ int i=0; do {i++;} while (i<8);

10) Welche Aussagen treffen für Konstruktoren zu?
- ☐ sie sind immer ‘void’
- ☐ sie gibt es immer genau einmal in einer Klasse
- ☐ sie tragen den selben Namen wie die Klasse
- ☐ es gibt sie nicht in Java

# Appendix B.4
# Task-Descriptions for User-Study

*Task 1*:

Karl studiert im dritten Semester Maschinenbau. Er lebt ein typisches Studentenleben, ist ein engagierter Student und fühlt sich trotzdem oft zeitlich unter Druck gesetzt. Insbesondere in den Zeiten vor Prüfungen und Abgaben steigt ihm die Arbeit über den Kopf. Blickt Karl auf einen Monat zurück, weiß er oft nicht genau, wie er seine Zeit genutzt hat. Er hat die Sorge, sich seine Zeit nicht effektiv genug einzuteilen.

Helfen Sie Karl in dem Sie eine App entwerfen, mit der Karl seinen Zeitaufwand besser bewerten und planen kann.

*Task 2*:

Lisa studiert im vierten Semester Informatik. Sie studiert gerne und engagiert, besucht allerdings häufig Kurse, in denen sie Aufgaben in Gruppenarbeit bearbeiten

muss. Oft kennt Lisa ihre Mit-Studenten im Kurs nur wenig, so dass die Gruppenzusammensetzung recht willkürlich verläuft…

Oft hat Lisa die Erfahrung gemacht, dass nicht alle Gruppenmitglieder gleich engagiert mitarbeiten, oder sich in ihren Fähigkeiten stark unterscheiden. Arbeitet Lisa in einer ungünstigen Gruppen-Konstellation, ist ihr persönlicher Arbeitsaufwand in der Gruppenarbeit sehr hoch und die abgegebenen Ergebnisse der Arbeit sind trotzdem nicht so gut, wie sich Lisa dies wünschen würde.

Helfen Sie Lisa und anderen Studenten indem Sie eine App entwerfen, in der sich Studenten registrieren und in Profilen verwalten können. In den Profilen könnten individuelle Fähigkeiten der Studenten wiedergegeben, Kursbesuche und Erfahrungen berichtet werden. Vielleicht könnte eine solche App auch helfen, spontane Arbeitsgruppen zu bilden.

*Task 3*:

Maria studiert im siebten Semester Elektrotechnik. Sie studiert gerne und engagiert, was sich auch in Ihren Leistungen wiederspiegelt. Vor den Prüfungen erstellt Maria zahlreiche Karteikarten, in denen sie Faktenwissen sammelt, welches sie für die Prüfung auswendig parat haben muss. Marias großer Alptraum ist es, Ihre wertvolle Kartensammlung vor einer Prüfung zu verlieren. Außerdem würde Maria gerne ihre wertvollen Fakten-Sammlungen mit anderen Studenten teilen, diskutieren und abgleichen.

Entwerft eine App, die Maria und ihren Mitstudenten hierbei helfen könnte!

# Appendix B.5
# Final Questionnaire

Paper- Based Prototyping (rein auf Papier basierendes Prototyping)

- ☐ Davon hatte ich vorher noch nie etwas gehört
- ☐ Hatte schon davon gehört, wusste aber nicht genau was das ist
- ☐ Ich wusste was das ist, hatte es aber zuvor noch nie angewendet
- ☐ Ich hatte die Methode schon einmal für ein Projekt genutzt
- ☐ Ich fühlte mich schon zuvor mit der Methode sehr vertraut

Axure Prototyping (das klassische Computer-Programm)

- ☐ Davon hatte ich vorher noch nie etwas gehört
- ☐ Hatte schon davon gehört, wusste aber nicht genau was das ist
- ☐ Ich wusste was das ist, hatte es aber zuvor noch nie angewendet
- ☐ Ich hatte die Methode schon einmal für ein Projekt genutzt
- ☐ Ich fühlte mich schon zuvor mit der Methode sehr vertraut

Blended Prototyping (der kombinierte Einsatz von Papier und Computer in einem Tabletopcomputing - Aufbau)

- ☐ Davon hatte ich vorher noch nie etwas gehört
- ☐ Hatte schon davon gehört, wusste aber nicht genau was das ist
- ☐ Ich wusste was das ist, hatte es aber zuvor noch nie angewendet
- ☐ Ich hatte die Methode schon einmal für ein Projekt genutzt
- ☐ Ich fühlte mich schon zuvor mit der Methode sehr vertraut

# Appendix B.6
# Categories for Assessing the Solutions' Ideation

*Ideation-Categories TASK 1*:

| Kategorie | | Implementierte Idee |
|---|---|---|
| Rückblick auf vergangene Zeit | Darstellung | Tortendiagramm |
| | | Balkendiagramm |
| | | Kennzahlen (Anzahl Termine, Summe der Stunden) |
| | Optimale Zeiteinteilung | von der App vorgegeben |
| | Bewertung der vergangenen Zeit | Infotext der App |
| | | Smileys |
| | | Vorschläge für die Zukunft |
| | fehlende Zeiten | durch Kalendereinträge oder Informationen aus anderen Apps ergänzen |
| | | wird als eigene Kategorie "ungenutzte Zeit" bewertet |
| Planung der Zeit | Darstellung | Kalender |
| | | Liste |
| | | Matrix (Eisenhower Schema) |
| | | Zeitstrahl des Tagesablaufs |
| | Termine Eintragen | Zuordnung der Termine zu vorgegebenen Kategorien |
| | | Zuordnung zu eigenen Kategorien |
| | | Aktiviäten bei Beginn eintragen und Zeit stoppen |
| | | Einteilung des geplanten Gesamtzeitaufwands vor Deadlines |
| | Prioritäten | Prioritäten für Termine angeben |
| | Planungshilfe | Warnung bei Konfliktterminen |
| | | Warnung bei einseitiger Zeitplanung (zu viel in einer Kategorie) |
| | Einhalten von Terminen | Reminder vor Deadline |
| | | Nachträgliche Abfrage ob Termine eingehalten wurden |
| | Anpassung auf den Nutzer | intelligente Einteilung der Zeiten durch das System |

*Ideation-Categories TASK 2*:

| Kategorie | | Implementierte Idee |
|---|---|---|
| Gruppenfunktionen | Gruppentreffen | App ermöglicht spontane Gruppentreffen |
| | | App ermöglicht regelmäßige Gruppentreffen |
| | | einmaliges Gruppentreffen |
| | Gruppe finden/gründen | Möglichkeit vorhanden, bestehende Gruppen zu finden und beizutreten |
| | Personensuche | erweiterte Personensuche (filtern nach Studiengang, Fähigkeiten, Zeit,…) |
| | passende Gruppenmitglieder finden | Vorschläge zu passenden Gruppenmitgliedern durch die App… anhand von Fähigkeiten/ Eigenschaften |
| | | Vorschläge zu passenden Gruppenmitgliedern durch die App… anhand von Terminmöglichkeiten |
| | | Warnung bei Personen mit negativen Eigenschaften |
| | Interaktion mit Gruppenmitgliedern | private Nachrichten |
| | | Gruppenchat/Forum |
| | | Dateien hochladen |
| | | Abstimmungen |
| | | Skypekonferenz |
| | | Interaktion mit dem Dozent |
| | Gruppenaufgaben | To-Do Liste mit Verantwortungszuweisung |
| | | Deadlines |
| | Verwaltung der Gruppen | Veknüpfung der Gruppen mit Unikursen |
| | | Gruppentreffen werden im Kalender angezeigt |
| Unikurse | Übersicht | Übersicht der Unikurse integriert |
| | Abgabe | Abgaben von Arbeiten durch die App möglich |
| persönliche Fähigkeiten/ Vorlieben | Selbsteinschätzung | belegte Kurse/Fachkenntnisse/Sprachkenntnisse |
| | | Fähigkeiten, Erfahrungen |
| | | Softskills, Eigenschaften |
| | | Schwächen |
| | Gruppenpräferenz | präferierte Gruppengröße |
| | | bevorzugte Rolle in der Gruppe |
| | | bevorzugtes Gruppentreffen (online/ offline) |
| | Eingabemöglichkeit | freie Texteingabe |
| | | mit Skala (z.B. Sterne, 1–5, …) |
| | | Checkboxen |
| | | Schieberegler |
| | Bewertung anderer | Möglichkeit die Eigenschaften anderer Nutzer zu bewerten |
| zeitliche Verfügbarkeit | Eintragen | eigene Verfügbarkeit im Profil eintragen |
| | | feste Termine in die Gruppe eintragen |
| | | flexibel in der Gruppe vorschlagen (abstimmen) |

*Ideation-Categories TASK 3*:

| Kategorie | | Implementierte Idee |
|---|---|---|
| Karteikarten | erstellen | Karteikarten eintippen |
| | | Karteikarten abfotografieren |
| | | Karteikarten anderer Nutzer in eigene Sammlung integrieren |
| | einseitig/zweiseitig | einseitig |
| | | zweiseitig |
| | lernen | Abfrage möglich (Rückseite duch Klick, o.ä.) |
| | | vorlesen lassen |
| | | Antwort eintippen, System vergleicht mit Lösung |
| | Reihenfolge | zufällige Reihenfolge möglich |
| | | von der App anhand des Lernstatus generiert (Karteikartensystem) |
| | Lernstatus | Nutzereingabe ob Antwort gewusst oder nicht |
| | | Karteikarten markiert/kommentiert werden |
| | Auswertung | Überblick über Lernstatus |
| | | Prognose des weiteren Lernaufwands |
| | Interaktion mit Freunden | Karteikarten mit bestimmten Personen teilen (freigeben) |
| | | automatischer Zugriff auf die Sammlungen aller Nutzer |
| | | Karteikarten kommentieren/diskutieren |
| | | Karteikarten/Sammungen bewerten |
| | | Chat |

# References


1. Adelmann R, Langheinrich M (2009) SPARK rapid prototyping environment—mobile phone development made easy. In Intelligent interactive assistance and mobile multimedia computing, vol 225–237. Springer, Berlin. Retrieved from doi:10.1007/978-3-642-10263-9_20
2. Dolenc A, Mäkelä I (1996) Rapid prototyping from a computer scientist's point of view. Rapid Prototyping J 2(2):18–25. doi:10.1108/13552549610128198
3. Ahmed M, Farag A (2005) Nonmetric calibration of camera lens distortion: differential methods and robust estimation. IEEE Trans Image Proc 14(8):1215–1230
4. Alvarado C, Davis R (2004) SketchREAD: a multi-domain sketch recognition engine. In: Proceedings of the 17th annual ACM symposium on user interface software and technology, ACM, pp 23–32. doi:10.1145/1029632.1029637
5. Amant RS, Horton TE, Ritter FE (2007) Model-based evaluation of expert cell phone menu interaction. ACM Trans Comput-Hum Interact 14(1):1. doi:10.1145/1229855.1229856
6. Apted T, Kay J, Quigley A (2006) Tabletop sharing of digital photographs for the elderly. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, pp 781–790
7. Bähr B (2010) A surface-augmented paper-prototyping system for mobil applications. Diplomarbeit, Technische Universität Berlin, Berlin
8. Bähr B (2011) Early design stage interface prototypes on mobile devices. Studienarbeit; Technische Universität Berlin
9. Bähr B (2013) Rapid creation of sketch-based native android prototypes with blended prototyping. Mobile HCI 2013—workshop on prototyping to support the interaction designing in mobile application development (PID-MAD 2013)
10. Bähr B (2015) Towards a requirements catalogue for prototyping tools of mobile user interfaces. In: Marcus A (ed) Design, user experience, and usability: users and interactions. Springer International Publishing, Berlin, pp 495–507. Retrieved 12 April 2016 from http://link.springer.com/chapter/10.1007/978-3-319-20898-5_48
11. Bähr B, Kratz S, Rohs M (2010) A tabletop system for supporting paper prototyping of mobile interfaces. "PaperComp" workshop, UbiComp 2010 Copenhagen, Denmark. ACM
12. Bähr B, Möller S (2016) Blended prototyping. In: Gengnagel C, Nagy E, Stark R (eds) Rethink! prototyping. Springer International Publishing, Berlin, pp 129–160. Retrieved 24 Feb 2016 from http://link.springer.com/chapter/10.1007/978-3-319-24439-6_9
13. Bähr B, Neumann S (2013) Blended prototyping design for mobile applications. In: Rethinking prototyping: proceedings of the design modelling symposium Berlin 2013. epubl, pp 68–80
14. Bailey BP, Biehl JT, Cook DJ, Metcalf HE (2008) Adapting paper prototyping for designing user interfaces for multiple display environments. Pers Ubiquit Comput 12 (3):269–277. doi:10.1007/s00779-007-0147-2




B. Bähr, *Prototyping of User Interfaces for Mobile Applications*,
T-Labs Series in Telecommunication Services,
DOI 10.1007/978-3-319-53210-3

15. Ballagas R, Memon F, Reiners R, Borchers J (2007) iStuff mobile: rapidly prototyping new mobile phone interfaces for ubiquitous computing. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, pp 1107–1116. doi:10.1145/1240624.1240793
16. Basu A, Licardie S (1993) Modeling fish-eye lenses. In: Proceedings of the 1993 IEEE/RSJ international conference on intelligent robots and systems '93, IROS '93, vol 3, pp 1822–1828
17. Battocchi A, Pianesi F, Tomasini D et al (2009) Collaborative puzzle game: a tabletop interactive game for fostering collaboration in children with autism spectrum disorders (ASD). In: Proceedings of the ACM international conference on interactive tabletops and surfaces, ACM, pp 197–204. doi:10.1145/1731903.1731940
18. Bederson B, Grosjean J, Meyer J (2004) Toolkit design for interactive structured graphics. IEEE Trans Softw Eng 30(8):535–546
19. Bähr B (2012) Thoughts on blended prototpying. In: Prototype! physical, virtual hybrid, smart tackling new challenges in design and engineering. Form + Zweck, pp 88–100
20. Beyer H, Holtzblatt K (1997) Contextual design: defining customer-centered systems (interactive technologies). Morgan Kaufmann, Burlington
21. Bias RG (1994) The pluralistic usability walkthrough: coordinated empathies. In: Usability inspection methods. Wiley, Hoboken, pp 63–76. Retrieved 10 Feb 2016 from http://dl.acm.org/citation.cfm?id=189211
22. Blanchard BS (2012) System engineering management. Wiley, Hoboken
23. Blanchard BS, Fabrycky WJ (2013) Systems engineering and analysis. Pearson Education Limited, Upper Saddle River
24. Blinn JF (1990) The ultimate design tool. IEEE Comput Graph Appl 10(6):90–92
25. Bonanni L, Vaucelle C, Lieberman J, Zuckerman O (2006) PlayPals: tangible interfaces for remote communication and play. In: CHI'06 extended abstracts on human factors in computing systems, ACM, pp 574–579. Retrieved 12 Feb 2016 from http://dl.acm.org/citation.cfm?id=1125572
26. Borchers J, Ringel M, Tyler J, Fox A (2002) Stanford interactive workspaces: a framework for physical and graphical user interface prototyping. IEEE Wireless Commun 9(6):64–69. doi:10.1109/MWC.2002.1160083
27. Brave S, Ishii H, Dahley A (1998) Tangible interfaces for remote collaboration and communication. In: Proceedings of the 1998 ACM conference on computer supported cooperative work, ACM, pp 169–178. doi:10.1145/289444.289491
28. Brewster S (2002) Overcoming the lack of screen space on mobile computers. Pers Ubiquit Comput 6(3):188–205
29. Buchenau M, Suri JF (2000) Experience prototyping. In: Proceedings of the 3rd conference on designing interactive systems: processes, practices, methods, and techniques, ACM, pp 424–433. doi:10.1145/347642.347802
30. Buisine S, Besacier G, Najm M, Aoussat A, Vernier F (2007) Computer-supported creativity: evaluation of a tabletop mind-map application. In: Proceedings of the 7th international conference on engineering psychology and cognitive ergonomics. Springer, Berlin, pp 22–31
31. Buxton B (2007) Sketching user experiences: getting the design right and the right design. Morgan Kaufmann, Burlington
32. Card S, Newell A, Moran T (1983) The psychology of human-computer interaction. L. Erlbaum Associates Inc. Retrieved 7 Dec 2015 from http://portal.acm.org/citation.cfm?id=578027
33. Carson SH, Peterson JB, Higgins DM (2005) Reliability, validity, and factor structure of the creative achievement questionnaire. Creativity Res J 17(1):37–50. doi:10.1207/s15326934crj1701_4

34. Cherubini M, Venolia G, DeLine R, Ko AJ (2007) Let's go to the whiteboard: how and why software developers use drawings. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, pp 557–566. doi:10.1145/1240624.1240714
35. Consolvo S, Walker M (2003) Using the experience sampling method to evaluate ubicomp applications. IEEE Pervasive Comput 2(2):24–31
36. Cook DJ, Bailey BP (2005) Designers' use of paper and the implications for informal tools. In: OZCHI '05, pp 1–10. Retrieved 2 June 2014 from http://dl.acm.org/citation.cfm?id=1108368.1108402
37. Coyette A, Kieffer S, Vanderdonckt J (2007) Multi-fidelity prototyping of user interfaces. In: Baranauskas C, Palanque P, Abascal J, Barbosa SDJ (eds) Human-computer interaction—INTERACT 2007. Springer, Berlin, pp 150–164. Retrieved 4 Sept 2013 from doi:10.1007/978-3-540-74796-3_16
38. Coyette A, Vanderdonckt J (2005) A sketching tool for designing anyuser, anyplatform, anywhere user interfaces. In: Costabile MF, Paternò F (eds) Human-computer interaction—INTERACT 2005. Springer, Berlin, pp 550–564. Retrieved 4 Sept 2013 from doi:10.1007/11555261_45
39. Dalmasso I, Datta SK, Bonnet C, Nikaein N (2013) Survey, comparison and evaluation of cross platform mobile application development tools. In: Wireless communications and mobile computing conference (IWCMC), 2013 9th international, pp 323–328. doi:10.1109/IWCMC.2013.6583580
40. Davis RC, Saponas TS, Shilman M, Landay JA (2007) SketchWizard: wizard of Oz prototyping of pen-based user interfaces. In: Proceedings of the 20th annual ACM symposium on user interface software and technology, ACM, pp 119–128. doi:10.1145/1294211.1294233
41. Derboven J, De Roeck D, Verstraete M, Geerts D, Schneider-Barnes J, Luyten K (2010) Comparing user interaction with low and high fidelity prototypes of tabletop surfaces. In: Proceedings of the 6th Nordic conference on human-computer interaction: extending boundaries, ACM, pp 148–157. doi:10.1145/1868914.1868935
42. Devernay F, Faugeras O (2001) Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured environments. Mach Vis Appl 13(1):14–24
43. Dickinson TL, McIntyre RM (1997) A conceptual framework for teamwork measurement. Team Perform Assess Measur 19–43
44. Diefenbach S, Chien WC, Lenz E, Hassenzahl M (2013) Prototypen auf dem Prüfstand. Bedeutsamkeit der Repräsentationsform im Rahmen der Konzeptevaluation. i-com Zeitschrift für interaktive und kooperative Medien 12(1):53–63
45. Diefenbach S, Hassenzahl M, Eckoldt K, Laschke M (2010) The impact of concept (re) presentation on users' evaluation and perception. In: Proceedings of the 6th Nordic conference on human-computer interaction: extending boundaries, ACM, pp 631–634. doi:10.1145/1868914.1868991
46. Dorst K, Cross N (1995) Protocol analysis as a research technique for analysing design activity. In: Design engineering technical conferences, pp 563–570
47. Dow SP, Glassco A, Kass J, Schwarz M, Schwartz DL, Klemmer SR (2010) Parallel prototyping leads to better design results, more divergence, and increased self-efficacy. ACM Trans Comput-Hum Interact 17(4):18:1–18:24. doi:10.1145/1879831.1879836
48. Henry Duh, Gerald Tan, and Vivian Chen (2006) Usability evaluation for mobile device: a comparison of laboratory and field tests. In: MobileHCI '06: proceedings of the 8th conference on human-computer interaction with mobile devices and services. ACM Press, pp 181–186. Retrieved from doi:10.1145/1152215.1152254
49. Ferrise F, Bordegoni M, Lizaranzu J (2010) Product design review application based on a vision-sound-haptic interface. In: Nordahl R, Serafin S, Fontana F, Brewster S (eds) Haptic and audio interaction design. Springer, Berlin, pp 169–178. Retrieved 7 Dec 2015 from doi:10.1007/978-3-642-15841-4_18

50. Filipowicz A (2006) From positive affect to creativity: the surprising role of surprise. Creativity Res J 18(2):141–152. doi:10.1207/s15326934crj1802_2
51. Fischer G (2001) User modeling in human-computer interaction. User Model User-Adap Interact 11(1–2):65–86. doi:10.1023/A:1011145532042
52. Bonadiman F (2016) Enhancing the interaction space of a tabletop computing system to design paper prototypes for mobile applications
53. Goel V (1995) Sketches of thought (Bradford Books). The MIT Press, Cambridge
54. Görner R (1973) Ergebnisse und Probleme aus empirischen psychologischen Untersuchungen der Konstrukteurstätigkeit. In: Vortrag zum 8. Symposium "Methoden und Mittel zur Rationalisierung der Konstruktionsarbeit". Dresden, TU Dresden, Sektion Elektrotechnologie und Feingerätetechnik
55. Gutwin C, Greenberg S (2000) The mechanics of collaboration: developing low cost usability evaluation methods for shared workspaces. In: Proceedings of the 9th IEEE international workshops on enabling technologies: infrastructure for collaborative enterprises, IEEE Computer Society, pp 98–103. Retrieved 30 Apr 2015 from http://dl.acm.org/citation.cfm?id=647068.715651
56. Hacker W (1999) Konstruktives Entwickeln als Tätigkeit: Versuch einer Reinterpretation des Entwurfsdenkens (design problem solving). Sprache & Kognition 18(3–4):88–97
57. Haller M, Brandl P, Leithinger D, Leitner J, Seifried T, Billinghurst M (2006) Shared design space: sketching ideas using digital pens and a large augmented tabletop setup. In: Pan Z, Cheok A, Haller M, Lau RWH, Saito H, Liang R (eds) Advances in artificial reality and tele-existence. Springer, Berlin, pp 185–196. Retrieved 24 Jan 2014 from doi:10.1007/11941354_20
58. Hassenzahl M, Burmester M, Koller F (2003) AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität. In: Szwillus G, Ziegler J (eds) Mensch & Computer 2003. Vieweg+Teubner Verlag, pp 187–196. Retrieved 18 May 2015 from doi:10.1007/978-3-322-80058-9_19
59. Hendry DG, Mackenzie S, Kurth A, Spielberg F, Larkin J (2005) Evaluating paper prototypes on the street. In: CHI '05 extended abstracts on human factors in computing systems, ACM, pp 1447–1450
60. Hilliges O, Terrenghi L, Boring S, Kim D, Richter H, Butz A (2007) Designing for collaborative creative problem solving. In: Proceedings of the 6th ACM SIGCHI conference on creativity and cognition, ACM, pp 137–146. doi:10.1145/1254960.1254980
61. Jung HG, Lee YH, Yoon PJ, Kim J (2010) Radial distortion refinement by inverse mapping-based extrapolation
62. Holzmann C, Vogler M (2012) Building interactive prototypes of mobile user interfaces with a digital pen. In: Proceedings of the 10th Asia pacific conference on computer-human interaction, ACM, pp 159–168. doi:10.1145/2350046.2350080
63. Hong JI, Landay JA (2000) SATIN: a toolkit for informal ink-based applications. In: Proceedings of the 13th annual ACM symposium on user interface software and technology, ACM, pp 63–72
64. Hornecker E, Marshall P, Dalton NS, Rogers Y (2008) Collaboration and interference: awareness with mice or touch input. In: Proceedings of the 2008 ACM conference on computer supported cooperative work, ACM, pp 167–176. doi:10.1145/1460563.1460589
65. Horst W (2011) Supportive tools for collaborative prototyping. In: Nordes 0, 4. Retrieved 2 June 2014 from http://www.nordes.org/opj/index.php/n13/article/view/147
66. Hosseini-Khayat A, Seyed T, Burns C, Maurer F (2011) Low-fidelity prototyping of gesture-based applications. In: Proceedings of the 3rd ACM SIGCHI symposium on engineering interactive computing systems, ACM, pp 289–294. doi:10.1145/1996461.1996538
67. Houde S, Hill C (1997) What do prototypes prototype? In Handbook of human-computer interaction, 2nd edn. North-Holland, pp 367–382

68. Hunter S, Maes P (2008) WordPlay: a table-top interface for collaborative brainstorming and decision making. In: Proceedings of IEEE tabletops and interactive surfaces, pp 2–5
69. Hupfer S, Cheng LT, Ross S, Patterson J (2004) Introducing collaboration into an application development environment. In: Proceedings of the 2004 ACM conference on computer supported cooperative work, ACM, pp 21–24. doi:10.1145/1031607.1031611
70. Landay JA, Myers BA (2009) Just draw it! Programming by sketching storyboards
71. Israel JH (2010) Hybride Interaktionstechniken des immersiven Skizzierens in frühen Phasen der Produktentwicklung. Retrieved from doi:10.14279/depositonce-2490
72. John B, Vera A, Matessa M, Freed M, Remington R (2002) Automating CPM-GOMS. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, pp 147–154. doi:10.1145/503376.503404
73. Kaikkonen A, Kallio T, Kekäläinen A, Kankainen A, Cankar M (2005) Usability testing of mobile applications: a comparison between laboratory and field testing. J Usability Stud 1 (1):4–16
74. Kazi RH, Chevalier F, Grossman T, Zhao S, Fitzmaurice G (2014) Draco: bringing life to illustrations. In: CHI '14 extended abstracts on human factors in computing systems, ACM, pp 579–582. doi:10.1145/2559206.2574769
75. Kelley T (2002) The art of innovation. Profile Business, City
76. Kieras D (2003) The human-computer interaction handbook. In: Jacko JA, Sears A (eds) L. Erlbaum Associates Inc., Hillsdale, pp 1139–1151. Retrieved 7 Dec 2015 from http://dl.acm.org/citation.cfm?id=772072.772143
77. Kjeldskov J, Skov MB, Als BS, Høegh RT (2004) Is it worth the hassle? Exploring the added value of evaluating the usability of context-aware mobile systems in the field. In: Mobile human-computer interaction—MobileHCI 2004. Springer, Berlin, pp 529–535. Retrieved from doi:10.1007/978-3-540-28637-0_6
78. Klemmer S, Newman M, Farrell R, Meza R, Landay JA (2000) A tangible difference: participatory design studies informing a designers' outpost. In: Workshop on shared environments to support face-to-face collaboration. Retrieved from http://dub.washington.edu:2007/projects/outpost/CSCWWorkshopSubmission.pdf
79. Klemmer SR, Everitt KM, Landay JA (2008) Integrating physical and digital interactions on walls for fluid design collaboration. Hum Comput Inter 23(2):138–213. doi:10.1080/07370020802016399
80. Klemmer SR, Newman MW, Farrell R, Bilezikjian M, Landay JA (2001) The designers' outpost: a tangible interface for collaborative web site. In: Proceedings of the 14th annual ACM symposium on User interface software and technology, ACM, pp 1–10. doi:10.1145/502348.502350
81. Koivisto EMI, Suomela R (2007) Using prototypes in early pervasive game development. In: Proceedings of the 2007 ACM SIGGRAPH symposium on Video games, ACM, pp 149–156. doi:10.1145/1274940.1274969
82. Kordon F, Luqi A (2002) An introduction to rapid system prototyping. IEEE Trans Softw Eng 28(9):817–821
83. Korhonen H, Paavilainen J, Saarenpää H (2009) Expert review method in game evaluations: comparison of two playability heuristic sets. In: Proceedings of the 13th international MindTrek conference: everyday life in the ubiquitous era, ACM, pp 74–81. doi:10.1145/1621841.1621856
84. Kratz S, Rohs M (2009) Unobtrusive tabletops: linking personal devices with regular tables
85. Kulkarni SV (1998) Survey of creativity models. Technical report ASU/DAL/IG/98-1, Arizona State University
86. Landay JA (1996) SILK: sketching interfaces like krazy. In: Conference companion on human factors in computing systems, ACM, pp 398–399. doi:10.1145/257089.257396
87. Landay JA, Myers BA (1995) Interactive sketching for the early stages of user interface design. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM Press/Addison-Wesley Publishing Co., pp 43–50

88. Landay JA, Myers BA (1996) Sketching storyboards to illustrate interface behaviors. In: Conference companion on human factors in computing systems: common ground, ACM, pp 193–194
89. Landay JA, Myers BA (2001) Sketching interfaces: toward more human interface design. Computer 34(3):56–64
90. Richard Landis J, Koch GG (1977) The measurement of observer agreement for categorical data. Biometrics 33(1):159–174. doi:10.2307/2529310
91. Leichtenstern K, André E (2010) MoPeDT: features and evaluation of a user-centred prototyping tool. In: Proceedings of the 2nd ACM SIGCHI symposium on engineering interactive computing systems, ACM, pp 93–102. doi:10.1145/1822018.1822033
92. Lewis C, Polson PG, Wharton C, Rieman J (1990) Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, pp 235–242. doi:10.1145/97243.97279
93. Lin J, Landay JA (2008) Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, pp 1313–1322. doi:10.1145/1357054.1357260
94. Lin J, Newman MW, Hong JI, Landay JA (2000) DENIM: finding a tighter fit between tools and practice for web site design. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, pp 510–517. doi:10.1145/332040.332486
95. Lumsden J, MacLean R (2008) A comparison of pseudo-paper and paper prototyping methods for mobile evaluations. In: Meersman R, Tari Z, Herrero P (eds) On the move to meaningful internet systems: OTM 2008 workshops. Springer, Berlin, pp 538–547. Retrieved July 23, 2013 from http://link.springer.com/chapter/10.1007/978-3-540-88875-8_77
96. Maloney J, Resnick M, Rusk N, Silverman B, Eastmond E (2010) The scratch programming language and environment. Trans Comput Educ 10(4):1–15
97. McCurdy M, Connors C, Pyrzak G, Kanefsky B, Vera A (2006) Breaking the fidelity barrier: an examination of our current characterization of prototypes and an example of a mixed-fidelity success. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, pp 1233–1242. doi:10.1145/1124772.1124959
98. McGee-Lennon MR, Ramsay A, McGookin D, Gray P (2009) User evaluation of OIDE: a rapid prototyping platform for multimodal interaction. In: Proceedings of the 1st ACM SIGCHI symposium on engineering interactive computing systems, ACM, pp 237–242. doi:10.1145/1570433.1570476
99. Miller GA (1956) The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol Rev 63(2):81–97. doi:10.1037/h0043158
100. Morris MR, Paepcke A, Winograd T (2006) TeamSearch: comparing techniques for co-present collaborative search of digital media. In: Proceedings of the first IEEE international workshop on horizontal interactive human-computer systems, IEEE Computer Society, pp 97–104
101. Muller MJ (1991) PICTIVE—an exploration in participatory design. In: CHI '91 proceedings of the SIGCHI conference on human factors in computing systems. ACM, New York, pp 225–231. Retrieved from http://dl.acm.org/citation.cfm?id=108896
102. Nagai Y, Noguchi H (2002) How designers transform keywords into visual images. In: Proceedings of the 4th conference on creativity & cognition, ACM, pp 118–125
103. Newell A (1994) Unified theories of cognition. Harvard University Press
104. Newman MW, Landay JA (2000) Sitemaps, storyboards, and specifications: a sketch of Web site design practice. In: Proceedings of the 3rd conference on designing interactive systems: processes, practices, methods, and techniques, ACM, pp 263–274
105. Newman MW, Lin J, Hong JI, Landay JA (2003) DENIM: an informal web site design tool inspired by observations of practice. Hum Comput Interact 18(3):259–324

106. Nielsen CM, Overgaard M, Pedersen MB, Stage J, Stenild S (2006) It's worth the hassle!: the added value of evaluating the usability of mobile systems in the field. In: NordiCHI '06, ACM, pp 272–280
107. Nielsen J (1992) The usability engineering life cycle. Computer 25(3):12–22
108. Nielsen J (1993) Usability engineering. Morgan Kaufmann
109. Nielsen J (1993) Iterative user-interface design. Computer 26(11):32–41
110. Nielsen J (1994) Usability inspection methods. In: Conference companion on Human factors in computing systems, ACM, pp 413–414. Retrieved 10 Feb 2016 from http://dl.acm.org/citation.cfm?id=260531
111. Nielsen J, Molich R (1990) Heuristic evaluation of user interfaces. In: Proceedings of the SIGCHI conference on human factors in computing systems, ACM, pp 249–256. doi:http://doi.org/10.1145/97243.97281
112. Norman D (1988) The psychology of everyday things. Basic Books
113. Norman D (2002) The design of everyday things. Basic Books
114. Olson G, Olson J (2000) Distance matters. Hum Comput Inter 15(2):139–178
115. Paterno F (2000) Model-based design and evaluation of interactive applications. Springer, Berlin
116. Piper B, Ratti C, Ishii H (2002) Illuminating clay: a 3-D tangible interface for landscape analysis. In: Proceedings of the SIGCHI conference on Human factors in computing systems: changing our world, changing ourselves, ACM, pp 355–362
117. Radatz J (1997) The IEEE standard dictionary of electrical and electronics terms. IEEE Standards Office
118. Reenskaug W, Reenskaug T, Lehne OA (1995) Working with objects: OORAM software engineering method. JA Majors, Greenwich
119. Regenbrecht H, Haller M, Hauber J, Billinghurst M (2006) Carpeno: interfacing remote collaborative virtual environments with table-top interaction. Virtual Real 10(2):95–107
120. Rekimoto J (1997) Pick-and-drop: a direct manipulation technique for multiple computer environments. In: Proceedings of the 10th annual ACM symposium on user interface software and technology, ACM, pp 31–39
121. Richter J, Thomas BH, Sugimoto M, Inami M (2007) Remote active tangible interactions. In: Proceedings of the 1st international conference on Tangible and embedded interaction, ACM, pp 39–42. doi:10.1145/1226969.1226977
122. Riesenbach R (1994) The Ontario telepresence project. In: Conference companion on human factors in computing systems, ACM, pp 173–176
123. Sachse P (2002) Idea materialis: Entwurfsdenken und Darstellungshandeln: über die allmähliche Verfertigung der Gedanken beim Skizzieren und Modellieren. Logos-Verlag
124. de Sá M, Carriço L (2006) Low-fi prototyping for mobile devices. In: CHI '06 extended abstracts on human factors in computing systems, ACM, pp 694–699. 10.1145/1125451.1125592
125. de Sá M, Carriço L (2008) Lessons from early stages design of mobile applications. In: Proceedings of the 10th international conference on human computer interaction with mobile devices and services, ACM, pp 127–136. doi:10.1145/1409240.1409255
126. de Sá M, Carriço L (2009) A mobile tool for in-situ prototyping. In: Proceedings of the 11th international conference on human-computer interaction with mobile devices and services, ACM, pp 20:1–20:4. doi:10.1145/1613858.1613884
127. de Sá M, Carriço L, Duarte L, Reis T (2008) A mixed-fidelity prototyping tool for mobile devices. In: Proceedings of the working conference on advanced visual interfaces, ACM, pp 225–232. doi:10.1145/1385569.1385606
128. Diefenbach S, Hassenzahl M (2011). Handbuch zur Fun-ni toolbox: user experience evaluation auf drei Ebenen. Retrieved 22 Dec 2015 from http://fun-ni.org/wp-content/uploads/Diefenbach+Hassenzahl_2010_HandbuchFun-niToolbox.pdf

129. Sauppé A, Mutlu B (2014) How social cues shape task coordination and communication. In: Proceedings of the 17th ACM conference on computer supported cooperative work & social computing, ACM, pp 97–108. doi:10.1145/2531602.2531610
130. Schön DA (1983) The reflective practitioner: how professionals think in action. Basic Books
131. Scott SD, Sheelagh M, Carpendale T, Inkpen KM (2004) Territoriality in collaborative tabletop workspaces. In: Proceedings of the 2004 ACM conference on computer supported cooperative work, ACM, pp 294–303
132. Segura VCVB, Barbosa SDJ (2013) UISKEI++: multi-device wizard of oz prototyping. In: Proceedings of the 5th ACM SIGCHI symposium on engineering interactive computing systems, ACM, pp 171–174. doi:10.1145/2480296.2480337
133. Shah JJ, Smith SM, Vargas-Hernandez N (2003) Metrics for measuring ideation effectiveness. Des Stud 24(2):111–134. doi:10.1016/S0142-694X(02)00034-0
134. Carpendale S, Miede A, Isenberg T et al (2010) Collaborative interaction on large tabletop displays
135. Shen C, Everitt K, Ryall K (2003) UbiTable: Impromptu face-to-face collaboration on horizontal interactive surfaces. In: UbiComp 2003: ubiquitous computing, pp 281–288. Retrieved from http://www.springerlink.com/content/uv4dwaye8dpjxck8
136. Shen C, Lesh NB, Vernier F, Forlines C, Frost J (2002) Sharing and building digital group histories. In: Proceedings of the 2002 ACM conference on computer supported cooperative work, ACM, pp 324–333
137. Shneiderman B (2007) Creativity support tools: accelerating discovery and innovation. Commun ACM 50(12):20–32. doi:10.1145/1323688.1323689
138. Shneiderman B, Plaisant C, Cohen M, Jacobs S (2013) Designing the user interface: Pearson New International Edition: strategies for effective human-computer interaction. Addison Wesley
139. Smutny P (2012) Mobile development tools and cross-platform solutions. In: 13th international carpathian control conference (ICCC), pp 653–656. doi:10.1109/CarpathianCC.2012.6228727
140. Snyder C (2003) Paper prototyping: the fast and easy way to design and refine user interfaces (The Morgan Kaufmann Series in Interactive Technologies). Morgan Kaufmann
141. Sommerville I (2010) Software engineering. Addison Wesley, Harlow
142. Spindler M, Stellmach S, Dachselt R (2009) PaperLens: advanced magic lens interaction above the tabletop. In: Proceedings of the ACM international conference on interactive tabletops and surfaces, ACM, pp 69–76
143. Streitz NA, Geißler J, Holmer T et al (1999) i-LAND: an interactive landscape for creativity and innovation. In: Proceedings of the SIGCHI conference on human factors in computing systems: the CHI is the limit, ACM, pp 120–127
144. Svanaes D, Seland G (2004) 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, ACM, pp 479–486. doi:10.1145/985692.985753
145. Szekely P (1994) User interface prototyping: tools and techniques. ICSE workshop on SE-HCI, 76–92. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.6764
146. Szekely P (1995) User interface prototyping: tools and techniques. In: Software engineering and human-computer interaction lecture notes in computer science. Springer, Berlin, pp 76–92. Retrieved from http://link.springer.com/chapter/10.1007/BFb0035808
147. Takano K, Shibata H, Omura K, Ichino J, Hashiyama T, Tano S (2012) Do tablets really support discussion?: comparison between paper, tablet, and laptop PC used as discussion tools. In: Proceedings of the 24th Australian computer-human interaction conference, ACM, pp 562–571. doi:10.1145/2414536.2414623

148. Tandler P, Prante T, Müller-Tomfelde C, Streitz N, Steinmetz R (2001) Connectables: dynamic coupling of displays for the flexible creation of shared workspaces. In: Proceedings of the 14th annual ACM symposium on User interface software and technology, ACM, pp 11–20
149. Tuddenham P, Davies I, Robinson P (2009) WebSurface: an interface for co-located collaborative information gathering. In: Proceedings of the ACM international conference on interactive tabletops and surfaces, ACM, pp 181–188
150. Tversky B, Suwa M, Agrawala M et al (2003) Sketches for design and design of sketches. In: Human behaviour in design. Springer, Berlin, pp 79–86. Retrieved 12 Apr 2016 from http://link.springer.com/chapter/10.1007/978-3-662-07811-2_9
151. Underkoffler J, Ishii H (1999) Urp: a luminous-tangible workbench for urban planning and design. In: Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit, ACM, pp 386–393
152. de Villiers JP, Wilhelm Leuschner F, Geldenhuys R (2008) Centi-pixel accurate real-time inverse distortion correction
153. Mitchell Waldrop M (2016) The chips are down for Moore's law. Nature 530(7589):144–147. doi:10.1038/530144a
154. Walker M, Takayama L, Landay JA (2002) High-fidelity or low-fidelity, paper or computer? Choosing attributes when testing web prototypes. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol 46, No. 5, pp 661–665. doi:10.1177/154193120204600513
155. Wallace JR, Scott SD, Stutz T, Enns T, Inkpen K (2009) Investigating teamwork and taskwork in single- and multi-display groupware systems. Pers Ubiquit Comput 13(8):569–581. doi:10.1007/s00779-009-0241-8
156. Walther-Franks B, Herrlich M, Karrer T et al (2012) Dragimation: direct manipulation keyframe timing for performance-based animation. In: Proceedings of graphics interface 2012, Canadian Information Processing Society, pp 101–108. Retrieved 27 July 2013 from http://dl.acm.org/citation.cfm?id=2305276.2305294
157. Wang D, Shen L, Wang H (2012) A collaborative sketch animation creation system on mobile devices. In: Proceedings of the ACM 2012 conference on computer supported cooperative work companion, ACM, pp 239–242. doi:10.1145/2141512.2141587
158. Ward TB, Smith SM, Finke RA (1999) Creative cognition. Retrieved 9 Feb 2016 from http://psycnet.apa.org/psycinfo/1998-08125-010
159. Weiser M (1995) The computer for the 21st century. In: Baecker RM, Grudin J, Buxton WAS, Greenberg S (eds) Human-computer interaction. Morgan Kaufmann Publishers Inc., pp 933–940
160. Wellner P (1993) Interacting with paper on the DigitalDesk. Commun ACM 36(7):87–96
161. Wilson AD, Sarin R (2007) BlueTable: connecting wireless mobile devices on interactive surfaces using vision-based handshaking. In: Proceedings of graphics interface 2007, ACM, pp 119–125
162. Wilson J, Rosenberg D (1988) Rapid prototyping for user interface design. In: Helander J (ed) Handbook for human-computer interaction. Elsevier Science Publishers, pp 859–875
163. Wong YY (1992) Rough and ready prototypes: lessons from graphic design. In: Posters and short talks of the 1992 SIGCHI conference on human factors in computing systems, ACM, pp 83–84. doi:10.1145/1125021.1125094
164. Wu M, Balakrishnan R (2003) Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In: Proceedings of the 16th annual ACM symposium on user interface software and technology, ACM, pp 193–202. doi:10.1145/964696.964718
165. Zufferey G, Jermann P, Lucchi A, Dillenbourg P (2009) TinkerSheets: using paper forms to control and visualize tangible simulations. In: Proceedings of the 3rd international conference on tangible and embedded interaction, ACM, pp 377–384
166. Balsamiq (2014) Rapid, effective, and fun wireframing software. Retrieved 3 June 2014 from http://www.balsamiq.com

167. Axure (2014) Interactive wireframe software & mockup tool. Retrieved 3 June 2014 from http://www.axure.com/
168. MockFlow (2014) Super-easy wireframing. Retrieved 3 June 2014 from http://www.mockflow.com/
169. POPAPP|POP—Prototyping on Paper (2014) POPAPP. Retrieved 28 May 2014 from https://popapp.in/
170. Add to cart interaction (2016) Marvel prototyping. Retrieved 10 Feb 2016 from https://marvelapp.com/2c67b1g?emb=1&exp=1
171. LimeSurvey (2014) The free and open source survey software tool! Retrieved 3 June 2014 from http://www.limesurvey.org/en/
172. Blockly (2016) Google developers. Retrieved 12 Feb 2016 from https://developers.google.com/blockly/