

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 words *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

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 its 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 loses its value after the testing, and can be literally speaking thrown away. For not wasting too 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 insights 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,

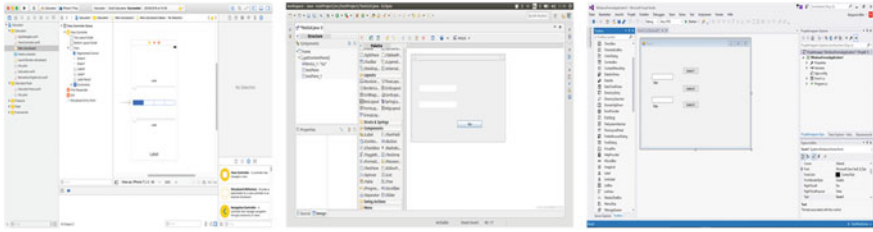


Fig. 2.1 Modern IDEs that provide visual GUI editors (from left to right Apple xCode, Eclipse, and Microsoft Visual Studio)

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 her¹ 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 accidentally 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

¹Remark 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-loading*, 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 conclusion 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 device's 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 interpreted version are displayed next to each other. Elements 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

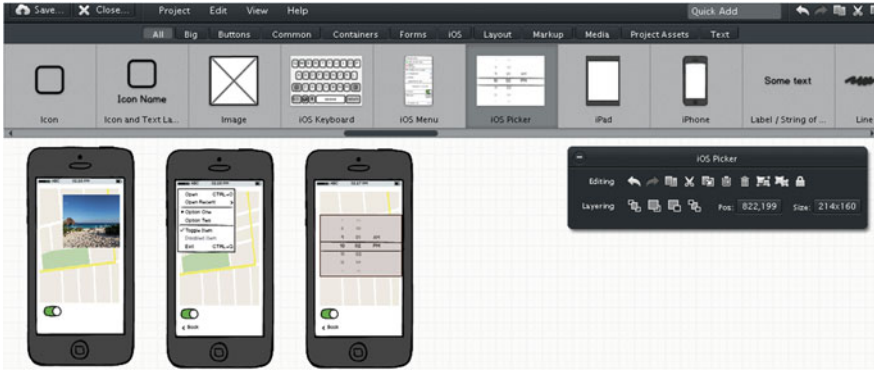


Fig. 2.2 Balsamiq editor in sketch-skin view

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.

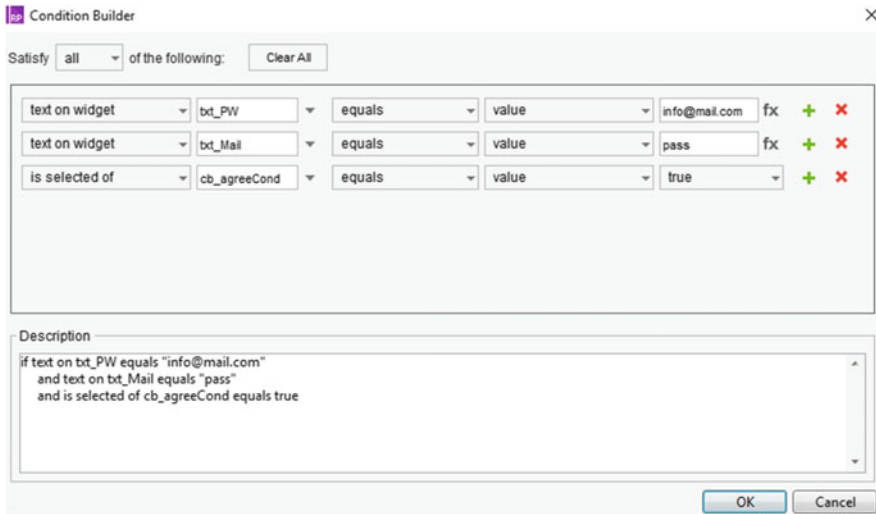


Fig. 2.3 Condition builder in Axure

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.

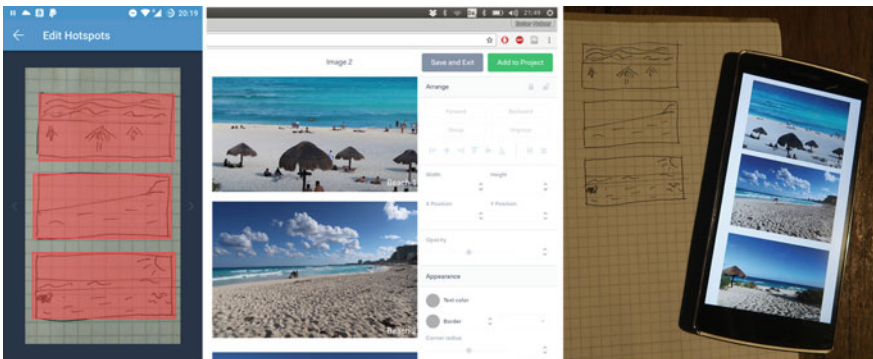


Fig. 2.4 Marvel mobile App (*left*), web-application (*middle*), running high-fi prototype (*right*)

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 a 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 be 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 be embedded into the interaction process, than in ordinary computer systems. Hilliges et al. [60] show this in a collaborative user study, where 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 HockeyApp⁴ 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

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

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

⁴www.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	SáPOP/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](#).

Prototyping of User Interfaces for Mobile Applications

Bähr, B.

2017, X, 162 p. 29 illus., 23 illus. in color., Hardcover

ISBN: 978-3-319-53209-7