Making Design Tangible in Software Development Projects

Lennart Hennigs

Abstract

User Experience (UX) Professionals often face the problem that their role and their contribution to software development projects are misunderstood. They are confronted with new challenges when they join agile or lean project teams. Over the following pages I will explain how UX professionals can make design tangible. I will describe how they can guide software development projects, how they can create a common understanding of the targeted User Experience and how they can involve others in the shaping of the design. After outlining the current challenges, I will explain how activities such as *Design Studios, Sketching* and *Prototyping* can be used to foster an understanding of the design rationale, and how artifacts like *Product Vision Statements*, *Design Tenets*, *Personas, User Scenarios, Wireframes* and *UI Flows* help to frame the design problem and document the design solution. Using these activities and artifacts will lead to smoother project operations and better results.

1 Introduction

User-Centered Design (UCD) and User Experience (UX) practices focus on creating products and services with a high level of usability. In the 'early days' of our discipline the main challenges were to create an understanding of UCD within companies and 'fight' to be included in the existing software development processes.

As our discipline matured, we moved away from simply recommending usability testing at the end of the product development cycle, when the product was almost done and we started to develop full User-Centered Design process models

L. Hennigs
Deutsche Telekom AG, Bonn, Germany
e-mail: Lennart.Hennigs@telekom.de

A. Maedche et al. (eds.), *Software for People*, Management for Professionals,
DOI 10.1007/978-3-642-31371-4_9, © Springer-Verlag Berlin Heidelberg 2012

that describe the various activities and documents needed to ensure user-centricity throughout software development projects (Mayhew 1999, Constantine & Lockwood 1999 and ISO 2011). It was during this time that our view of how usable products ought to be created was shaped, along with our view of our role of within software development. Simply focusing on *ease of use* it not sufficient, we need to think 'outside our own box'. Don Norman (1999) used the metaphor of the *three-legged stool* to describe the three factors (or 'legs') a successful product needs: Technology, Business and User Needs. He states that a product can only be successful if these three factors are equally taken into account. Cut short on one of them and a product will 'fall'.

Nowadays, most of the software development industry has a general understanding of the concept of usability. The term is known and UX practitioners don't need to explain their value every time they join a project. However our challenges did not diminish – they merely shifted.

Where we previously had to find our place within software development projects, we now have to rethink our way of working in line with new software development methodologies. The waterfall approach to software development[1] is less frequently used and more agile and lean software projects (Schwaber & Sutherland (2011); Poppendieck 2004) are carried out. Agile and lean development provides less time for upfront conception and design work. They are less structured and put more emphasis on self-organized, interdisciplinary teams that work autonomously. They suggest short development cycles and quick iterations, in which parts of the final product are being designed, developed, tested and deployed. We have to adjust our current practice to meet these new challenges. A step-by-step approach will not work. We need to offer a toolbox of activities and documents to meet the demands of agile software development approaches.

Questions we need to answer are:

- How can UX practitioners provide value to an interdisciplinary team?
- What activities are the best fit in an agile software development environment?
- How can you involve your team members in describing the targeted User Experience?
- What is the right document to communicate the design to your team?

Being able to work with and to provide value to an interdisciplinary team becomes even more important in an agile environment. We need to understand what the other people involved are contributing and how we can best support them. We need a set of activities and artifacts that can create and document the common understanding of the targeted user experience. And last but not least we need the other team members to contribute to our own activities.

Artifacts that allow different disciplines to share their knowledge are called *boundary objects*, first described by Susan Leigh Star & James R. Griesemer (1989). They are "scientific objects which both inhabit several intersecting social worlds [. . .] and satisfy the informational requirements of each of them." They are

[1] http://en.wikipedia.org/wiki/Waterfall_model

"both plastic enough to adapt the local needs and constraints of the several parties employing them, yet robust enough to maintain a common identity across sites."

The good news is I strongly believe that we already have the documents available that can serve as boundary objects for interdisciplinary teams communicating the design. We also have the skills to support and guide interdisciplinary teams. We are able to establish a common mindset for the problem domain. We can capture the design rationale in such a way that it can offer guidance throughout the software development process. With our focus on the user, on their tasks and on the context of use, we have a broader perspective of the problem domain than most other disciplines involved. Our discipline strongly embraces the idea of a designed user experience – an experience tailored to the user's needs and expectations. This is more than just addressing usability and ease-of-use issues: we are trying to delight our users with the products and service we design.

Activities like *Design Studios, Sketching* and *Prototyping* are easy to explain to and carry out with non-UX professionals. They are low-fidelity approaches, fast paced and allow members of a development team to contribute to the design of the product. Artifacts like *Product Vision Statements*, *Design Tenets*, *Personas, User Scenarios, Wireframes* and *UI Flows* are fairly easy to read and understand for non-UX professionals. They are more compact than typical software development documents (such as Requirement Specifications or Product Definition documents), but they capture the essence of the problem domain and the solution and foster a common understanding.

2 The Situation Today

Software development and design projects are *Wicked Problems* (Poppendieck 2002; Dorst 2003). They belong to a family of problems that share certain attributes that makes them hard (or wicked) to solve. Wicked problems are typically characterized by the involvement of different stakeholder with different views and priorities. Furthermore, the requirements of such a project are complex and interlinked. The problem is hard to describe and keeps changing while we are trying to solve it. Its solution will be unique – there is no precedent for it. There is no prior indication as to what an optimal solution will look like – there are no 'boxes to tick' while developing our solution. Only the end product will show how well it is suited solving the problem. However, there are some best practices that can ease the problem's 'wickedness'. John C. Camillus (2008) who analyzed company strategy creation recommends the following steps when faced with a wicked problem:

1. Define a common vision.
2. Document ideas and communicate.
3. Involve stakeholders.
4. Take small steps forward and evaluate and iterate.

As UX professionals we are already are doing these steps (to some degree) in our projects and day-to-day work. The following sections will discuss methods and documents fitting these recommendations and the benefit they provide during software development.

2.1 Define a Common Vision

There are two aspects of design relevant to software development projects: problem setting and problem solving. In other words, defining what it is we want to solve and how we approach it.

Since software development projects are wicked problems, agreeing what is to be achieved (setting the problem) is key. Here, the problem space is being explored: what constraints does the team want to place on themselves; what are their working assumptions, etc.?

2.1.1 The Product Vision Statement

To create a common goal you first need to establish a common understanding of the expectations and the requirements of the different parties involved in your project. You need to come to some form of agreement on the project scope and its targets. You need to define the criteria for the projects' success. You need to understand what the other parties can contribute and how you will be able to support each other. This is typically done in some form of kick-off meeting. One key artifact that should be created during a kick-off meeting is the *Product Vision Statement*. It describes the characteristics of the final product in a few sentences and explains the targeted user experience. It needs to be brief and to the point in order to make the statement easy to remember, easy to communicate and relate to.

A good example is the product vision statement of Metro, the design language of Windows Phone 7 shown in Fig. 1: "Metro is our design language. We call it Metro because it's modern and clean. It's fast and in motion. It's about content and typography. And it's entirely authentic." (Shum 2010)

Another way to create a vision statement is to use the elements described by Geoffrey Moore (2002) in his book "Crossing the Chasm": the target audience for the product, their needs, the product category, the key functionality and the major benefits, current practice/competition and key differentiators. These can be used to fill in the blanks in the following sentences "For _____ who are dissatisfied with_____. Our product is a _____ that provides _____ unlike_____, we have assembled_____."

2.1.2 Design Tenets

Ideally, after you have created the *Product Vision Statement* you should create a short set of guidelines (less than 10) describing the Product Vision in more depth. These guidelines are often referred to as *Design Principles* or *Design Tenets*. "Design principles are short, insightful phrases that act as guiding lights and support the development of great product experiences. Design principles enable you to be true to your users and true to your strategy over the long term." (Buley 2009)

The goal is to have a set of principles that can help when faced with design decisions. They should inspire the team and guide their decisions. They function as a beacon, highlighting how the user experience of the finished product ought to be. Good principles are specific to your project, concrete and non-ambiguous, catchy

Fig. 1 Product Vision Statement of Windows Metro (Copyright by Microsoft)

and describe differentiating properties (and not only a single feature) of your product (Saffer 2009; Anderson 2011; Spool 2001).

Good examples of design tenets are the principles behind the HTC Sense User Interface (UI) that was introduced on the HTC Hero mobile phone in 2009. With this HTC was the first company to offer a customized version of Google's Android mobile operating system and the first company to create a more user-centric and visually pleasing version of Android. Their Sense user interface was based on the following principles.[2]

- **Make it Mine**: Personalization needs to reach a level never before possible.
- **Stay Close**: Staying in touch with the people in your life means managing a variety of communication channels and applications.
- **Discover the Unexpected**: Many of the most memorable moments in your life are experienced, not explained.

These principles not only offer insights on the targeted user experience. They are also an example of how well defined design principles can be implemented and utilized because HTC used them to explain its product and features (see Fig. 2).

A frequent point of discussion is whether design tenets should be general statements like 'easy to use' or if they should be tailored to your project. I recommend the latter. If your design tenets are too general they become 'boilerplate phrases'. Of course, everyone wants to create a product that is 'easy to use'; no one would try to design something that didn't meet this criterion. Try to make your tenets specific to your product, so that they help you with the design decision you will face.

[2] http://www.youtube.com/watch?v=Kax24GN1458

Fig. 2 Screenshot from the HTC Hero Product Tour Video (Copyright by HTC, http://www.youtube.com/watch?v=kshGq8COSiM)

You can use general principles as a starting point to create your own, project-specific design tenets. Good examples of general design principles are the "10 Principles of Good Design" by Dieter Rams (1993). In addition, Human Interface Guidelines (HIG) contain general statements about the user experience for a specific platform, e.g. for iOS or Windows Phone 7 (Microsoft 2011; Apple 2011). For each general principle you need to ask yourself what it means for your product, your users, your context of use, your business, etc. By doing so you can extract specific rules tailored to your problem.

A frequently asked question is whether the product vision and the design tenets should be created prior to user research, or afterwards. The answer is: there is no best way. Usually it is advisable to do research first, before establishing your design idea but Don Norman (2011) states that reversing the sequence will also work: Create the design tenets and validate them afterwards.

2.2 Document Ideas and Communicate

Another recommended activity to solve a wicked problem is to document your ideas and communicate them frequently to your team and the involved stakeholders. These are the two key tasks for creating a consistent user experience – but also the key success factors. As Bill Buxton (2007) states: "Successful execution of a design depends on communication, and capturing the design rationale is an important component in this."

Why are these two factors so important? Because design (as an activity) consists of a large set of small decision that results in how the product or service that we create looks, feels and behaves. Thus these design decisions make up the product

experience – the perceived effect your product or service will have on the user. Being consistent in these small decisions creates a certain style that will be reflected in the product's experience. However, creating a consistent user experience is easier for a single individual than for a team: "The solo designer or artist produces works with this integrity subconsciously; he tends to make each micro decision the same way every time he encounters it." (Brooks 2010) That is why it is of utmost importance that a team has an agreed common vision and documents and communicates their ideas so that all team members consciously or unconsciously shape, share and adapt a common style in their design work.

2.2.1 Sketching

We commonly use diagrams and sketches for different purposes when working on a design. It can be said that sketching out our ideas and thoughts serves as a means for (visual) conversations we have with the problem, ourselves and others. We use it to frame the problem and to solve it. While sketching our thoughts, new ideas will emerge – that is why sketching is *generative*: sketching sparks new ideas in us as well as capturing the ideas we already have.

Another strength of sketching is that it is done on paper. As Sellen and Harper (2003) point out: "[Paper] will continue to predominate in activities that involve knowledge work, including browsing through information, reading and make sense of information; organizing and structuring and reminding of ideas; [. . .] and activities that involve showing and demonstrating ideas and actions to others." This is due to the *affordance* of paper (for details on the concept of affordance see Soegard 2008) – i.e. paper can easily be used for these types of human activity, more so than digital media.

Another advantage of sketches is that they are low-fidelity. Sketches are quick and easy to make, inexpensive and easy to dispose of (so you won't grow too attached to them), they suggest new ideas due to their ambiguity and don't offer too much detail. "Learning from sketches is based largely on the ambiguous nature of their representation. That is, they do not specify everything and lend themselves to, and encourage, various interpretations that were not consciously integrated into them by their creator." (Buxton 2007)

The following best practices are suggested by Brown (2011) to create better sketches: Initially list the information you have and you want to capture, make a first set of sketches, get some feedback, iterate your concepts and sketch out a different angle to the problem. Re-order elements to see where it leads, review your results with the input you started with and use conventions in your sketches to make the consistent and easier to understand. Use color sparingly and label your sketches to make them easy to read for others.

What do we sketch? We sketch the problem domain: We picture our understanding of the current situation – how things relate to each other, what steps the user currently needs to take with the current solution, etc. But we also sketch the solution: We picture the situation as we plan it to be. Our solution can be depicted in *Information Architecture* Diagrams, *Wireframes* or *User Interface Flows*. These different types of diagrams will be described later on.

While sketches encourage the discussion a designer has "with himself" they also foster the discussion among team members. Whether you use them in structured meetings such as *Design Studios*, which will be introduced later, or used 'ad-hoc' to discuss ideas, sketches often 'say' more than 1,000 words.

Depending on the concept you are trying to capture and how you sketch it (i.e. the type of diagram you use) your sketches will vary in fidelity. One dimension of fidelity is the level of granularity. A sketch could describe things from a 30,000 ft point of view or provide a micro view of a specific detail. An *Information Architecture Diagram*, for example, provides a high level view of the elements of an application or web site. The other dimension is sketching is the level of abstraction used. A sketch can roughly lay out an idea or be very detailed. Sometimes *Wireframes* contain only the general content of a page, and sometimes they are very specific and show the visual design and the interface elements to be used. Of course, this depends what you want to document in or communicate with your sketch.

2.2.2 Wireframes and User Interface Flows

Wireframes depict the layout of a user interface. They describe the structure, navigation, content and behavior of a single screen (or parts of it); its visual design is not shown. "The aim is to focus the team's attention and encourage conversation about what a screen does, not what it looks like." (Brown 2011) Wireframes are one of the key deliverables of UX professionals because they "are a means of documenting the features of a product, as well as the technical and business logic that went into those features, with only a veneer of visual design [. . .]. They are the blueprints of a product." (Saffer 2009)

A *User Interface Flow* is a set of wireframes visualizing an 'interaction path'; highlighting what interface elements were used through the flow (e.g. a mouse click or gesture on a touch interface) and how the system responded (e.g. with animations, transitions, pop-up dialogs, the next screen) while working with the system. Sometimes conditions are visualized as well (e.g. error cases) to showcase important variations of the user interface.

A wireframe shows the relationship and the hierarchy of the page elements. Questions a wireframe should answer are: What are the main components of the user interface? How is it organized? What information is important? What information is secondary?

A wireframe highlights the navigational structure of a UI. It should explain the navigational elements that are used, how the user knows where he is, how he can he navigate away from the current screen and what his options are.

A wireframe explains the content of a user interface. Related questions include: What type of content is needed? How is it displayed? Can the user interact with it? What content is important and how is this communicated?

A wireframe showcases the behavior of a UI. It highlights the interactive elements, their relationship to each other and how the user gets feedback from the product.

Wireframes usually contain annotations to explain certain details. They sometime highlight a design decision, explain how to interact with an element, explain its

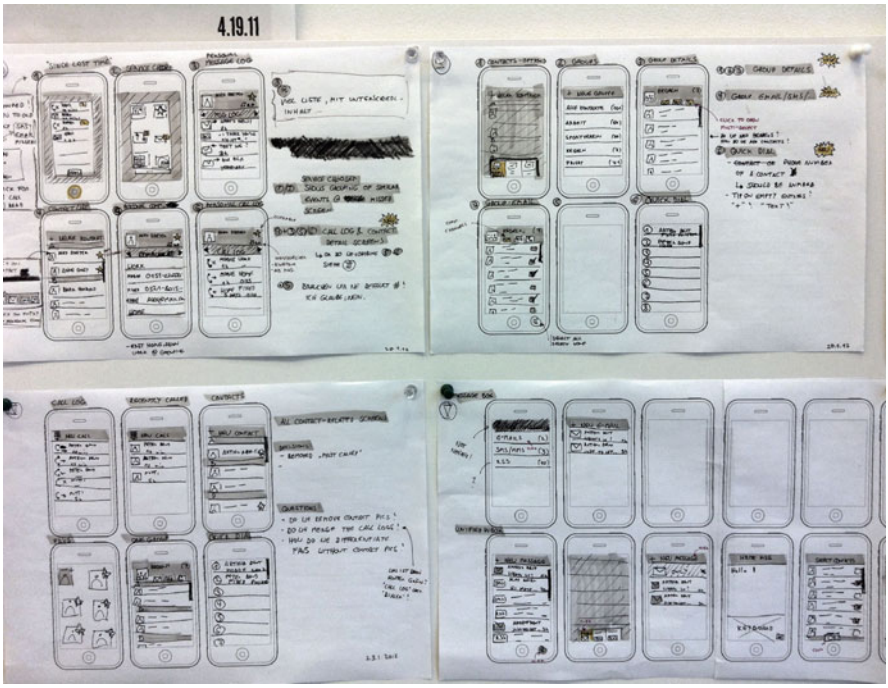

Fig. 3 Low-fidelity wireframes (of a mobile application). Copyright by the author

content, and point out a special case or an open issue in the wireframe. Points of interest are numbered and explained in a sidebar or highlighted by arrows, depending on personal preference.

Wireframes can be created with different levels of fidelity. Their visual, functional and content fidelity can be differentiated.[3] The simplest wireframes are typically sketches (as shown in Fig. 3) done on paper. Low-fidelity wireframes are fast to create and to discard. UX professionals create them to try out different variations of the user interface and to see if an idea could work (Boersma 2010; Johnson 2011). If a set of sketches is 'stable' enough (i.e. they correctly answer the questions the UX practitioners created them for) they are transferred into higher fidelity wireframes. These are usually created with dedicated wireframing tools and will be closer to the final design of the product. Ward (2008) showcases wireframes of the same screen in different stages of completion and different levels of fidelity.

2.2.3 Information Architecture Diagrams

Information Architecture Diagrams (IA diagrams) provide the 30,000 ft point of view of an application or web site. They visualize the different areas of a product

[3] Different styles of wireframes can be viewed at http://wireframes.tumblr.com/

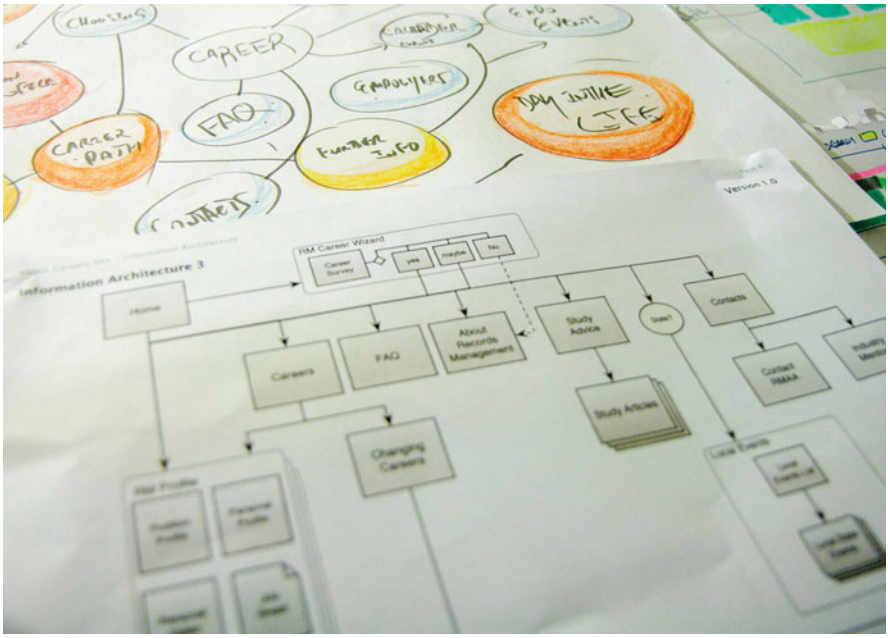

Fig. 4 Information architecture diagram (Photo by Gary Barber, http://www.flickr.com/photos/cannedtuna/4853380320/in/photostream/)

and how they are connected to each other. They provide an overview of the scale of the overall solution and indicate what paths a user can take within the product. They are the 'floor plan' of your solution. For web sites IA diagrams are often called *Sitemaps*.

Together with wireframes they are also a key deliverable of UX professionals because IA diagrams provide a high-level view of the solution, while wireframes show the content of the different areas (Fig. 4).

Different notations exist for IA diagrams, such as Jesse James Garrett's "Visual Vocabulary" (2002) or Jacob Linowski's "Interactive Sketching Notation" (2011).

Information Architecture diagrams consist of a set of nodes connected to each other via arrows showing the possible navigation paths. Sometimes different types of nodes are used (e.g. for different types of content) and similar items belonging together (e.g. part of the same page or area) are grouped. Nodes are labeled to explain their purpose and make them distinct. Sometimes flowchart elements (such as diamonds visualizing decision points) are also added for clarification.

2.2.4 Personas

Personas are an established means of documenting knowledge about the targeted user groups, first described by Alan Cooper (1999) in his book "The Inmates are running the Asylum" (see also Pruitt and Adlin 2010; Mulder and Yaar 2006).

Fig. 5 Persona descriptions (Photo by Gary Barber, http://www.flickr.com/photos/cannedtuna/4852756417/in/photostream/)

A persona is a human-friendly format for user-research related facts. It is a description of an intended user of the solution to be designed. It distills the information about a specific user group into a fictitious user profile.

Persona descriptions usually include some background information about the persona (name, a picture showing the person, age, gender, and a product-related quote), the persona's needs and requirements and often a scenario describing the persona's typical day or an event where the solution we are trying to design would benefit them. These are called User Scenarios, which will be described in the next section (Fig. 5).

The strength of personas is that they offer a tangible and accessible format for user research findings – they literally give mere facts 'a human face.' People will relate to them as if they were real people, and when discussing design decisions they will, for example, say: "If we do it this way we won't help Peter." Personas leave a longer lasting impression than a simple bullet-point list stating facts.

That's why it is essential that the persona is based on user-research findings. If it is not, it is a work of fiction and it will not provide value. A persona needs to capture the real needs and requirements of your intended users. Otherwise the persona will not be of any use in guiding your design decisions. If you create a persona based on assumptions, you need to validate them, similar to design tenets.

The format and layout of personas are also very important. Sometimes personas are printed on posters, or poster boards are created of the personas to make them visible and accessible to people. The documents should be easy to read, understand

and remember, because you want your team to refer to them frequently. Last but not least, all information that is not useful for making design decisions should be removed from a persona, e.g. stating that a persona owns a dog is clutter unless you are designing a pet-related product.

2.2.5 User Scenarios

User Scenarios are stories that describe a sequence of events leading up to an outcome. They are less formal than *Use Cases*, which are used to capture requirements, describing an interaction sequence and all its variations (Cockburn 2000).

User scenarios usually include some hints on the motivation, knowledge and capabilities of the persona. They sometimes include tools and objects the persona uses. They "provide insight into the reasons and motivations for those events. Stories that accompany personas often describe something about their activities or experiences." (Quesenbery and Brooks 2010) They offer insights into the context of use as well as the personas' goals and motivations.[4]

Scenarios can be used for different purposes. If a scenario describes the current situation, they are used to define and capture the problem (the problem setting). If a scenario describes the vision of how the interaction will be with the future system, it describes the solution. Rosson and Carrol (2002) refer to the first type as scenarios as *Problem Scenarios* and to the second as *Design Scenarios*. In addition, scenarios can be used to describe a specific context of use or illustrate a current shortcoming or a pain-point of the existing solution. Scenarios should be only about one or two paragraphs long and describe a single topic.

Just like *Design Tenets* and *Personas, User Scenarios* require input and verification through user research because they are tools for capturing user insights and design decisions.

2.3 Involve Stakeholders

Due to their role UX practitioners are always 'caught in the middle' between the different stakeholders involved in a software development project. We negotiate with Product Managers about features and priorities, we discuss how the solution should look and behave with Developers, we talk to Marketing to understand their targeted customers, and we join System Analysts on site visits, etc. We therefore often have a broader view of the various complexities and interdependencies of a project.

Interdisciplinary work poses the following challenges to UCD practitioners:

- We are always under-represented within projects. Close cooperation with the other stakeholders and involving them in our own work is therefore a necessity for UCD practitioners.

[4] An example of a user scenario combined with some persona information can be found at: http://www.flickr.com/photos/rosenfeldmedia/4459979060/

- Usability is known as a term, its value is understood but its place within the project (not only after the development is complete) within the project is often misunderstood as is the impact it will have on the project (UX practitioners will discuss requirements, challenge design decisions, etc.)

That is why we need to make the other parties involved aware of the consequences of design decisions being taken during the development of a product.

2.3.1 Design Studio

A well-suited method of involving the other parties in the creation of a product is the *Design Studio* (Ungar 2008; Evans 2011; Lindstrom 2011). This approach is borrowed from Industrial Design and Architecture and offers a structured approach to problem solving and innovating. It essentially involves a meeting in which the participants sketch different design options, discuss their sketches and agree on a direction to move forward. It is independent of the software development model used – it works both for waterfall or agile processes. As Ungar (2008) states: "The design studio is a collaborative workshop that fits well within the timeframes Agile software development practices while incorporating the benefits of UCD research." It is an iterative design and critique process where non-UX professionals can participate. Design Studio sessions are usually hosted by a moderator who keeps track of the time, as each activity is intended to be completed quickly, within a rather short time frame. The critique sessions are also moderated.

A typical design studio session comprises of the following steps: First, the problem you want to solve is briefly described. Then, each participant is given some time to brainstorm ideas and sketch them out – e.g. several screens of a product or website or steps of a process. Afterwards, each participant presents their ideas and gets feedback from their peers. The feedback needs to point out the strengths of the presented concepts and highlight areas that still need improvement. The participants are then given time to improve their concepts. At the end, the best elements of each concept are selected and combined into a final concept (Fig. 6).

A Design Studio session combines several techniques to rapidly create and evaluate design alternatives. The result of (individual) brainstorming sessions are visualized and criticized by the team. This is the idea generation phase. Individuals then improve their best idea, which is ultimately combined into the best-fitting solution. This is the idea refinement phase.

The session allows non-UX practitioners to participate in concept creation, offering them first-hand experience of design. It facilitates the knowledge transfer among the participants. Non-UX professionals gain a better understanding of how UX practitioners work, and in return they have the chance to the other participants' points of view. It creates a common understanding of the design decisions taken by the team and the implications of these, and last but not least it supports the team's commitment to the design.

A variant of a Design Studio session uses sketchboards to showcase the concepts created during the session, and was introduced by Adaptive Path, a design consultancy firm. It puts a strong focus on idea generation and refinement and suggests a

Fig. 6 A design studio sketchboard. Copyright by the author

structured approach for the critique session. (For details see Schauer 2007; Harrelson and Buley 2008; Downes 2010.)

2.4 Take Small Steps and Evaluate and Iterate

As explained before, *Sketching* and *Design Studio* are inherent iterative activities. Artifacts such as *Wireframes* and *Information Architecture Diagrams* can be created as low-fidelity paper versions before recreating them in a high-fidelity digital format. "Sketches and prototypes are both instantiations of the design concept however they serve different purposes, and therefore are concentrated at different stages of the design process. Sketches are dominant the early ideation stages, whereas prototypes are more concentrated at the later stages where things are converging within the design funnel." (Buxton 2007)

2.4.1 Prototyping

Prototyping is the practice of creating something to test your assumptions and learning from its results. "Prototyping is practice for people who design and make things. It's not simply another tool for your design toolkit – it's a design philosophy." (Warfel 2009)

Prototyping needs to be iterative, because each prototype shapes and improves your understanding of the problem and the solution domain. With prototypes you refine your design step by step. Unlike written requirements prototypes are able to

show and not only *tell* how parts of the solution behave. You can create prototypes to evaluate only a small aspect of the problem, or to look at the broad picture instead.

In their simplest forms, your wireframe sketches can serve as paper prototypes you evaluate by yourself, with colleagues or even with users (Snyder 2003). You want to get feedback and new ideas about the design problem you are facing. If your project is more advanced you might create high fidelity wireframes and prototypes of the user interface to gather more feedback on different aspects of your design.

Just as with wireframes you can choose different levels of fidelity for your prototypes. Again the dimensions are: visual, functional and content fidelity. Prototypes with a low level of visual fidelity contain sketches of the user interface, not showing the visual design. Prototypes with a low level functional fidelity will consist of a set of still screens; higher fidelity prototypes will offer interactive elements. Prototypes with a low level content fidelity can contain blind text ('Lorem ipsum'); higher fidelity version should showcase real content.

Prototypes should be created for the more complex aspects of your solution. "Good candidates for prototyping include complex interactions, new functionality and changes in workflow, technology or design." (Cerejo 2010) The most used functionality should also be prototyped – a good rule of thumb offers the Pareto Principle: What is 20 % of the functionality that is going to be used 80 % of the time?

Since prototyping is supposed to be done repeatedly and happen quickly you should try not to spend too much time on polishing the details of your prototypes. Your peers and test-users will understand that the prototype is not the real solution. Last but not least don't try to prototype the full solution. Prototypes are there to demonstrate a behavior or functionality you want to explore. Keeping this in mind will keep your prototypes' scope smaller and you'll be able to create them faster.

3 Summary

The previous sections described a set of low-fidelity activities and artifacts to capture the User Experience of a product. As these are easy to do and create by UX professionals, they also foster collaboration with non-UX people and create a shared understanding of and responsibility for the design and the User Experience of the product.

But the interesting question is: why do they work?

One reason is that they are people-friendly. Humans are visual creatures. *Sketching* lets us explore the problem and the solution space visually; it allows us to generate new ideas and see new connections. The same is true for *Prototyping*. We can show and see how we envision the solution (or parts of it), and test it out to gather feedback. We can show our sketches and prototypes to colleagues and team members to foster a shared understanding and learn their point of view. We are able to validate our assumptions and learn from our prototypes. We can later discard them easily because creating them did not cost too much effort. Humans are also

social creatures. *Design Studio* sessions foster teamwork and collaboration and create a common understanding of the design.

Another reason can be found in the book "Made to Stick" (2007) by Chip and Dan Heath. It describes six key characteristics to make ideas and concepts understandable and memorable. 'Sticky' ideas are simple, unexpected, concrete, credible, emotional and they tell a story. All of the techniques and artifacts described above share some of these traits. The *Product Vision Statement* and the *Design Tenets* are 'simple', in the sense that they describe the core design rationale of the solution. They are concrete and often emotional. *Personas* give a 'human face' to user research data, they become credible, while *User Scenarios* use stories to communicate and frame the design problem and the solution.

References

Anderson, S. (2011). Principles to build by. *IA Summit 2010*. Available online at http://www.slideshare.net/stephenpa/design-principles-to-build-by. Accessed 29 Jan 2012.

Apple Inc. (2011). *iOS human interface guidelines*. Available online at https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf. Accessed 29 Jan 2012.

Boersma, P. (2010). *Good design faster*. Available online at http://www.slideshare.net/pboersma/good-design-faster-at-design-by-fire-2010. Accessed 29 Jan 2012.

Brooks, F. P. (2010). *The design of design*. Boston: Pearson Education.

Brown, D. M. (2011). *Communicating design: Developing web site documentation for design and planning* (2nd ed.). Berkeley: New Riders.

Buley, L. (2009). *Design principles in a nutshell*. Available online at http://www.adaptivepath.com/ideas/d120209. Accessed 29 Jan 2012.

Buxton, B. (2007). *Sketching user experiences*. San Francisco: Morgan Kaufmann.

Camillus, J. C. (2008). Strategy as a wicked problem. *Harvard Business Review*, May 2008.

Cerejo, L. (2010). *Design better and faster with rapid prototyping*. Available online at http://www.smashingmagazine.com/2010/06/16/design-better-faster-with-rapid-prototyping/. Accessed 29 Jan 2012.

Cockburn, A. (2000). *Writing effective use cases*. Boston: Addison-Wesley.

Constantine, L. L., Lockwood, L. (1999). Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design, Reading, MA: Addison-Wesley.

Cooper, A. (1999). *The inmates are running the asylum*. Indianapolis: Sams.

Dorst, K. (2003). The problem of design problems. In N. Cross & E. Edmonds (Eds.), *Expertise in design* (pp. 135–147). Sydney: Creativity and Cognition Studio Press.

Downes, J. (2010). *Using sketchboards to design great user interfaces quickly*. Available online at http://www.boxuk.com/blog/using-sketchboards-to-design-great-user-interfaces. Accessed 29 Jan 2012.

Evans, W. (2011). *Introduction to design studio methodology*. Available online at http://uxmag.com/articles/introduction-to-design-studio-methodology. Accessed 29 Jan 2011.

Garrett, J. J. (2002). *A visual vocabulary for describing information architecture and interaction design*. Available online at http://www.jjg.net/ia/visvocab/. Accessed 29 Jan 2012.

Harrelson, D., & Buley, L. (2008). *Sketchboards and prototypes*. Available online at http://www.slideshare.net/ugleah/sketchboards-prototypes-presentation. Accessed 29 Jan 2012.

Heath, C., & Heath, D. (2007). *Made to stick: Why some ideas survive and others die*. New York: Random House.

ISO (2010). *ISO 9241–210:2010: Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems,* Switzerland.

Johnson, J. (2011). *Close photoshop and grab a pencil: The lost art of thumbnail sketches*. Available online at http://designshack.net/articles/inspiration/close-photoshop-and-grab-a-pencil-the-lost-art-of-thumbnail-sketches/. Accessed 29 Jan 2011.
Lindstrom, J. (2011). *Design studio: The good, the bad and the science*. Available online at http://www.uxbooth.com/blog/design-studios-the-good-the-bad-and-the-science/. Accessed 29 Jan 2012.
Linowski, J. (2011). *Interactive sketching notation*. Available online at http://www.linowski.ca/sketching. Accessed 29 Jan 2012.
Mayhew, D. (1999). *The usability engineering lifecycle*. San Francisco, CA: Morgan Kaufmann.
Microsoft (2011). *Windows user experience interaction guidelines*. Available online at http://msdn.microsoft.com/en-us/library/windows/desktop/aa511258.aspx. Accessed 29 Jan 2011.
Moore, G. A. (2002). *Crossing the chasm (Revised Edition)*. New York: Harper Business.
Mulder, S., & Yaar, Z. (2006). *The user is always right: A practical guide to creating and using personas for the web*. Berkeley: New Riders.
Norman, D. A. (1999). *The invisible computer: Why good products can fail, the personal computer is so complex, and information appliances are the solution*. Cambridge, MA: MIT Press.
Norman, D. A. (2011). *Act first, do the research later*. Available online at http://www.core77.com/blog/columns/act_first_do_the_research_later_20051.asp. Accessed 29 Jan 2012.
Poppendieck, M. (2002). Wicked projects. In *Software Development Magazine*. Available online at http://drdobbs.com/184414851. Accessed 29 Jan 2012.
Poppendieck, M. (2004). An introduction to lean software development. Available online at http://www.leanessays.com/2004/06/introduction-to-lean-software.html. Accessed 29 Jan 2012.
Pruitt, J., & Adlin, T. (2010). *The essential persona lifecycle: Your guide to building and using personas*. San Francisco, CA: Morgan Kaufmann.
Quesenbery, W., & Brooks, K. (2010). *Storytelling for user experience*. Brooklyn: Rosenfeld Media.
Rams, D. (1993). Ten principles for good design. Available online at http://www.vitsoe.com/en/gb/about/dieterrams/gooddesign. Accessed 29 Jan 2012.
Rosson, J. M. & Carroll J. M. (2002). Usability Engineering: Scenario-Based Development of Human-Computer Interaction, San Francisco , CA: Morgan Kaufmann.
Saffer, D. (2009). *Design for interaction* (2nd ed.). Berkeley: New Riders.
Schauer, B. (2007). Sketchboards: discover better + faster UX solutions. Available online at http://www.adaptivepath.com/ideas/sketchboards-discover-better-faster-ux-solutions. Accessed 29 Jan 2012.
Schwaber, K., & Sutherland, J. (2011). The scrum guide. Available online at http://www.scrum.org/scrumguides/. Accessed 29 Jan 2012.
Sellen, A. J., & Harper, R. (2003). *The myth of the paperless office*. Cambridge, MA: MIT Press.
Shum, A. (2010). Designing windows phone 7 series. *MIX 10, Las Vegas*. Available online at http://channel9.msdn.com/events/MIX/MIX10/CL14. Accessed 29 Jan 2012.
Soegard. (2008). Affordances. Available online at http://www.interaction-design.org/encyclopedia/affordances.html. Accessed 29 Jan 2012.
Spool, J. (2001). Creating great design principles: 6 counter-intuitive tests. Available online at http://www.uie.com/articles/creating-design-principles. Accessed 29 Jan 2012.
Star, S. L., & Griesemer, J. R. (1989). Institutional ecology, 'translations' and boundary objects. *Social Studies of Sciences, 19*(3), 387–420.
Synder, C. (2003). *Paper prototyping: The fast and easy way to define and refine user interfaces*. San Francisco: Morgan Kaufmann.
Ungar, J. (2008). The design studio: Interface design for agile teams. *Agile 2008 conference*, IEEE Computer Society, Washington.
Ward, J. (2008). Sketches, wireframes and CSS. Available online at http://jeff.io/posts/user-interface-wireframes. Accessed 29 Jan 2012.
Warfel, T. Z. (2009). *Prototyping: A practitioner's guide*. Brooklyn: Rosenfeld Media.