ingenieur
wissenschaften
htw saar

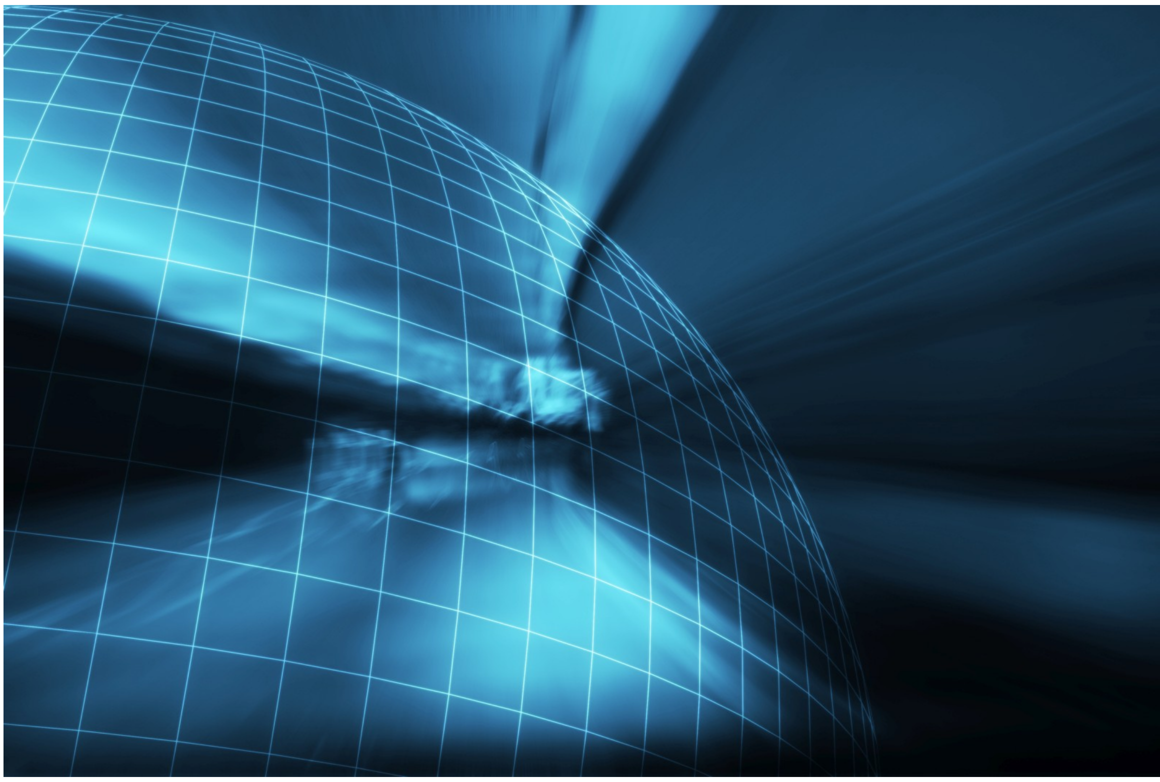
Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

STL SYSTEMTECHNIKLABOR

Means of Improving Usability on a Budget

Andreas Philippi

Technical Report – STL-TR-2016-04 – ISSN 2364-7167



stl.htwsaar.de

Technische Berichte des Systemtechniklabors (STL) der htw saar
Technical Reports of the System Technology Lab (STL) at htw saar
ISSN 2364-7167

Andreas Philippi: Means of Improving Usability on a Budget
Technical report id: STL-TR-2016-04

First published: September, 2016
Last revision: August, 2016
Internal review: André Miede

For the most recent version of this report see: http://stl.htwsaar.de/

Title image source: Flavio Takemoto (flaivoloka), http://www.freeimages.com/photo/1158737

htw saar – Hochschule für Technik und Wirtschaft des Saarlandes (University of Applied Sciences)
Fakultät für Ingenieurwissenschaften (School of Engineering)
STL – Systemtechniklabor (System Technology Lab)
Prof. Dr.-Ing. André Miede (andre.miede@htwsaar.de)
Goebenstraße 40
66117 Saarbrücken, Germany
http://stl.htwsaar.de

Means of Improving Usability on a Budget

Andreas Philippi
htw saar – Hochschule für Technik und Wirtschaft des Saarlandes
Seminar “Computer Science and Society”
Sommersemester 2016

***Abstract*—Despite their significance for an application’s success in the consumer market, usability aspects are often not considered important during the development of business software. But just like in the consumer market, bad user experience has the potential to leave employees frustrated and reduce their productivity. There are, however, easy ways in which every developer can enhance the usability of the interfaces he creates. Integrating those methods into the software development process is a cheap but efficient way of improving customer satisfaction.**

I. Introduction and Motivation

Visitors of a website decide within only ten seconds if the content of the website is relevant for them or not [1]. What is more, a study conducted by researchers at *Carleton University* in Canada showed that it takes just 50 milliseconds for them to judge the website’s visual appearance. On the Internet and in the software market in general, where a vast amount of different products for nearly every application leads to tough competition, those numbers show that the user’s first impression can determine the success of a product. But it is not only the first impression that counts. The same study found that the rate in which users leave the website slowly decreases over time, which means that if they already stayed for a while, it is likely that they will stay much longer [2]. Therefore, a well thought out user interface clearly constitutes a competitive advantage. A study which was conducted in 2011 by the *Institut für Mittelstandsforschung* among 160 German IT companies confirms this theory, concluding that increased usability of the software developed by the companies correlates with higher customer satisfaction and overall turnover [3, cmp. p. 232f].

In the consumer market, and particularly in the mobile application sector, this circumstance led to a large number of highly polished and user-centered applications. Due to the *consumerization* of business software, i.e. the “impact that consumer-originated technologies can have on enterprises” [4], the demand for usable software is brought into the companies by the employees: Besides the correlation between usability, customer satisfaction and turnover, the study by the Institut für Mittelstandsforschung also revealed that the usability of an application often is a relevant factor for companies when they make decisions on which software they buy. At the same time, when they were asked to rate aspects such as functionality, reliability, service and usability of the systems they currently use, the usability was rated clearly worse than the others. This leads to the conclusion that a gap between the desired quality of usability and the perceived reality of it exists [3, cmp. p. 139f].

Despite the potential that usability offers, the amount of resources dedicated to it during the development process is still comparatively small. Furthermore, especially smaller companies feel overwhelmed by the complexity and expensiveness of the common techniques [5, cmp. p. 5]. But improving the usability of an application does not necessarily have to be seen as a matter of ‘all or nothing’. As described in the following, a basic understanding of the psychology of interaction combined with simple techniques as well as a shift in perspective enables developers involved in a project to easily enhance the quality of the user interfaces they create.

II. Related Work

Several studies researched how measures to improve the usability of an application can be taken if constraints in time and financial resources limit the possible expenditure. In particular, those include the *Kompetenzzentrum Usability im Mittelstand (KUM)* [6] as well as the *Fraunhofer FIT* [5]. In their work, they elaborate on ways to implement a user-centered software development process for small and medium-sized businesses. The focus of this paper is slightly different: Instead of exploring cheap ways to transform the development itself, it proposes lightweight methods that individual developers can integrate into the existing process.

III. Fundamental Concepts of Interaction

Improving the usability of an application first requires a more precise understanding of what the term *usability* actually means. It can be described as the user-friendliness of software, or – more accurately – of the interfaces used by humans in order to interact with it [7, cmp. p. 304]. The scientific discipline concerned with the “design, evaluation and implementation of interactive computing systems for human use” [8] is called *Human Computer Interaction (HCI)*. It is related to the more general field of *Interaction Design* which was characterized by Preece et. al as “designing interactive products to support people in their everyday and working lives” [9, p. 6]. According to them, it focuses on effectiveness, efficiency, learnability and memorability from a user’s point of view as its main goals [9, cmp. p. 14]. In other terms, the product – or, in this case, the software – should not only serve its purpose in the best possible way from a technical standpoint. It should also enable users to quickly understand how to use it and remember that knowledge so they do not need to learn it again.

To achieve this, is is necessary to match the so-called *information architecture* of the system with the *mental model* of the people operating it. The information architecture is the actual design of the system that was created by its developers. It determines how the system works and how it presents itself to its users, e.g. through a *Graphical User Interface (GUI)* [10, cmp. p. 112]. Mental models, on the other hand, are images and patterns that we as humans form in our minds based on our previous experiences and our knowledge. We unconsciously apply them to both familiar and unfamiliar situations. This allows us to make assumptions about how things work and to predict which consequences our actions might have [11]. Therefore, a system will be perceived as intuitive and usable by its users if their mental model is similar to its information architecture. Matching them requires the consideration of several key factors like the goals and background of the users. Those arguably are highly dependent on the actual application and its target group, which makes it hard to find generic and reusable rules. Instead, it requires research specific to the project.

Nevertheless, there are general theories on interaction between a person and a thing that can help developers to get a better understanding of the common psychological aspects of human interaction, regardless of the concrete application. One of them was described by Don Norman as the *Gulfs of Execution and Evaluation*. On a high level, he considers interaction as a cycle of input and output between a user and a system. As shown in figure 1, it consists of seven *stages of action*, the first of which is the user's goal. It defines why he is interested in using the system in the first place and what he wants to accomplish in doing so. Since the goal triggers the user's interaction with the system, understanding it is crucial to the efficiency and effectiveness of the application.

In the second stage, the user looks at the interface of the system and intuitively tries to determine how it can help him to achieve his goal (1). After that, he breaks this general idea down into the individual steps he needs to perform (2) based on his mental model of the system. Afterwards, he actually executes them (3). Thus, this process of planning, specifying and executing is called the *Gulf of Execution*. Triggered by the input, the system starts working. It produces some kind of output which is again perceived by the user (4). In the next step, he interprets the perceived result (5) and compares it to his goal and the expectations about the result in his mind, i.e. his mental model, in order to evaluate what happened and whether he was successful (6). Those stages of action belong to the *Gulf of Evaluation*. [12, cmp. p. 38ff]

In this context, a system can be considered as *usable* if it takes adequate measures for each or at least most of the different stages of action. Obviously, this is again largely dependent on the target group, i.e. the users themselves. So how can developers learn about them if they have to deal with a tight budget?

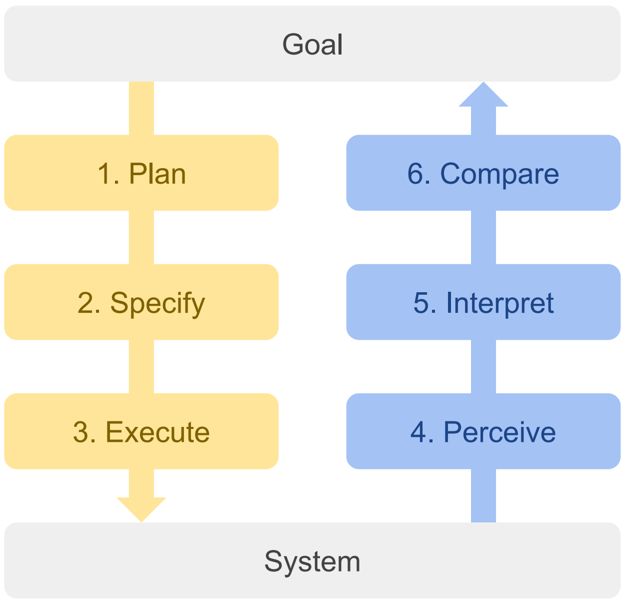


Figure 1. Gulfs of Execution and Evaluation [12, p. 41]

IV. Usability in the Software Development Process

As explained in the previous section, improving the usability of an application requires a good understanding of its users, but constraints in the project budget often limit the possibilities of acquiring such knowledge. Thus, lightweight methods are needed in order to incorporate at least a minimum of usability in the development of the product. For the following section, the terms *budget* and *lightweight* are defined as follows:

- *Budget* refers to the time and financial resources available during the project.
- Methods are characterized as *lightweight* if they can easily be integrated into the existing development process and neither require extensive knowledge nor consume noteworthy amounts of the budget.

To illustrate how and where lightweight usability methods can be applied during the development of software, this work assumes an iterative process that is comparable to the *Deming circle* [13, cmp. p. 16]. The Deming circle describes a process of continuous improvement by repeatedly going through the four phases depicted in figure 2:

1) *Plan*: The first phase contains both the analysis of the Status Quo and the requirements as well as the creation of a concept to satisfy them.
2) *Do*: The second step is to implement the concept developed in the previous phase.
3) *Check*: During the third phase, the implementation is evaluated. Among others, this includes running software tests and getting customer feedback.
4) *Act*: Finally, the problems found during the evaluation are fixed or added to the backlog for the next iteration.

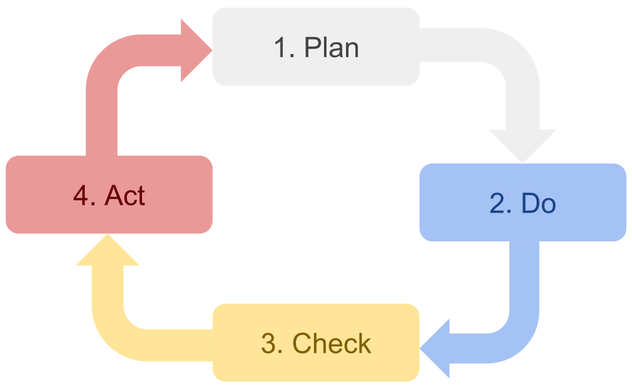
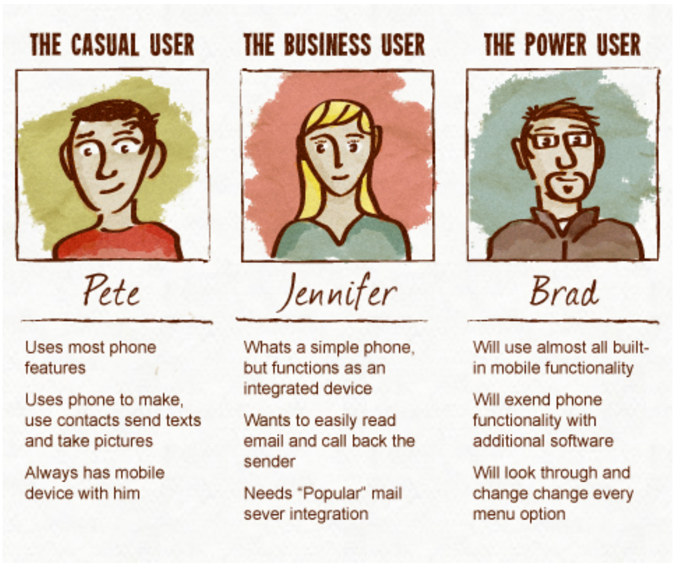


Figure 2. Deming cycle



THE CASUAL USER
THE BUSINESS USER
THE POWER USER
Pete
Jennifer
Brad
Uses most phone features
Uses phone to make, use contacts send texts and take pictures
Always has mobile device with him
Whats a simple phone, but functions as an integrated device
Wants to easily read email and call back the sender
Needs "Popular" mail sever integration
Will use almost all built-in mobile functionality
Will exend phone functionality with additional software
Will look through and change change every menu option

Figure 3. Example Personas [15]

The advantage of using this general process is that it can be applied to various common models. This could, for example, be agile models like *Scrum* as well as the more traditional *Spiral Model*. It could even be transferred to the *Waterfall Model*, except that the latter is missing the cyclic component. In the following, this paper proposes lightweight methods for each of those phases.

A. Plan

Since HCI is concerned rather with the users and their interaction with the system than with the technical details of the implementation, a well founded requirement specification and concept are crucial. Because of this, a large part of the typical activities of HCI take place in the planning phase. The primary intention at this point is to get to know the users and their context [9, cmp. p. 4], i.e. who they are, what task they need to accomplish, how it is structured from a functional point of view and how they did it so far. Usually the most straightforward way to get this information is to directly interact with the people that are concerned, for example by working with them for a while, conducting interviews and surveys or organizing workshops [10, cmp. p. 61]. But as efficient as those methods might be, they arguably require substantial amounts of budget and expertise. As a result, they are not always practicable.

Nevertheless, a typical requirement specification document usually already includes artifacts which turn out to be helpful. Besides the description of the functional requirements, particularly the *Use Case Diagrams* are of interest. They depict the different user types and what they need to do with the system. However, those depictions – also called *Actors* – are still quite abstract and it can be hard to imagine how they would act. Therefore, developers can apply a technique known as *Storytelling* which essentially aims at bringing the humans behind the Use Cases back into focus [14, cmp. p. 35]. To achieve this, the Actors in the diagram are replaced by so-called *Personas* as depicted in figure 3. A Persona is a fictional character that prototypically represents a group of users. It has its own distinct personality that can consist, among others, of a picture and demographic characteristics as well as a background story describing its goals and abilities [10, cmp. p. 78]. Analogously, each Persona has a *Scenario* for every Use Case that explains the specific reason why that use case exists, again embedded in a fictional story [9, cmp. p. 223].

By replacing the Actors with humans and Use Cases with stories, the originally impersonal Use Case Diagram comes to live in a way that causes our minds to empathize with those fictional characters. This effect can be compared to reading a book or watching a movie. Developers can now apply the Personas and their stories to help them with design decisions. Instead of wondering how someone would approach a certain problem, they can ask more concrete questions like 'Given his background and motivation, how would Jon Doe act in this situation?' and use their empathy to answer them.

The improved understanding of the users can now be used to create *Prototypes* of the interfaces with which they interact with the system. Those Prototypes give a first idea of how the application might look and behave, e.g. in terms of screen flows and layouts. Thus, they help finding difficulties and problems with the interface before the actual implementation [12, cmp. p. 227]. Especially if the budget is limited, they do not have to be very detailed and sophisticated: *Wireframes*, that will often be created and included in the functional specification anyway, or hand-drawn sketches can be done quickly and do not require much effort [16, cmp. p. 40]. Nevertheless, they too make their creator clarify the vague ideas he might already have and put some thinking into the layout instead of just improvising it during development. This alone already improves its quality.

When creating the Prototypes, there is often no need to reinvent the wheel: Relying on GUI design patterns serves a similar purpose as the design patterns of software engineering insofar as that they provide adaptable and often proved solutions to common problems. As many users nowadays are familiar with mobile, desktop and web applications, they already know common GUI and interaction patterns [17, cmp. p. 7ff] like Hamburger Menus, Master-Detail-Views and Swiping. They have incorporated them into their mental models and thus, the application will look and feel familiar to them even if they are using it for the first time. There are several resources providing extensive information on design patterns. The company behind the software *UXPin*, for example, offers a growing and up-to-date collection of e-books describing common GUI elements, layouts and trends[1]. If the application targets mobile devices running iOS[2] or Android[3], the interface guidelines of the respective platforms contain numerous examples and descriptions of best practices, too. Following them not only gives developers access to already existing knowledge but also makes it easier to achieve a look and feel that is consistent with other apps on the platform and avoids common pitfalls.

B. Do

As far as the topic of usability is concerned, the implementation phase is mainly the execution of the previously planned concept. If it does not already contain at least basic Prototypes of the GUI it is advisable to create some of them for the aforementioned reasons. Either way, they need to be transformed into an actual interface. However, while the Prototypes are likely to be quite low-level, developers need to consider a number of additional factors such as colors and typography, since they influence the perception and interpretation of the content of the GUI [10, cmp. p. 182]. But just like the design patterns, it is usually not necessary to start from scratch. Toolkits like *Material Design Lite*[4], *Bootstrap*[5] and the native libraries of the various mobile platforms already contain useful premade components as well as default colors and fonts. Again, applying them allows developers to benefit from existing knowledge and make their interfaces feel consistent.

C. Check

The third step is about validating the concept and its implementation. This includes two important parts: First of all, it is necessary to gather detailed feedback from the customer. Secondly, it is advisable to apply systematic evaluation methods to assess the quality of the system's user interface.

Besides careful planning, learning about the opinion of the customer regarding the results of the development is crucial. This is again a part that is usually covered anyway during the delivery, inspection and approval of the software. Just like in the typical development process, the advantage of using an iterative process instead of a linear one is obvious: The more frequently the product or parts of it are delivered or shown to the customer, the easier and cheaper it is to correct mistakes and make adjustments. Regarding the application's usability, this holds especially true since we as developers have, due to our technical background and involvement in the project, a perspective on the system that can be very different from the actual users. Ideally, gathering feedback also includes getting a small group of those people that will actually use the application to test it and let them state their opinion.

If that is not enough or not possible due to budget constraints, a 'better than nothing' alternative is a method called *Cognitive Walkthrough*. To perform such a Walkthrough, developers fall back to on the Personas that were created during the planning phase. Again, the empathy for the fictional characters helps them to shift their own perspective towards the user's point of view, making it easier to discover possible problems [14, cmp. p. 238]. Additionally, another quick technique of assessing the intuitiveness of an interface is the so-called *Hallway Testing*, during which a colleague not involved in the project is asked to have a look at the GUI and spontaneously describes his first thoughts, tries to solve simple tasks and describes what he thinks he is seeing [10, cmp. p. 226]. Similar to the Cognitive Walkthrough, the result provides, albeit not being as accurate as an actual user test, an easy way of discovering potential issues with minimal effort.

A more systematic approach to evaluating the user interface is known as *Heuristic Evaluation*. Originally described by Jakob Nielsen in 1990, the name stands for a test that assesses the quality of the interface on the basis of several different usability aspects. During the evaluation, the GUI is examined in closer detail regarding each of these heuristics. Among others, they include:

- *Visibility*: The user should always be able to determine the current status of the system, e.g. whether it is idle or computing something. This can, for example, be achieved by displaying feedback like progress bars and status messages.
- *Consistency*: The interface should follow conventions and use common patterns as well as appropriate, clear and uniform language, colors and icons, e.g. red for errors, exclamation marks for warnings.
- *Minimalism*: The GUI should be reduced to the minimum of what is necessary, allowing the user to focus on his goal and making it easier for him to decide which steps he needs to take.
- *Error prevention and recovery*: Ideally, the interface is constructed in a way that prevents errors, for example by immediately validating inputs and asking for confirmation before submitting them. If errors still occur, the application should help the user to recover from them by telling him what went wrong in understandable, non-technical terms, and offering information on how to fix the problem.

Each of them is assigned a value between 0 (no problems found) and 4 (fatal problem) depending on how likely it is that the problem occurs, if it is possible to avoid it and to which extent it affects the interaction. [18]

[1]https://www.uxpin.com/knowledge.html
[2]https://developer.apple.com/ios/human-interface-guidelines
[3]https://developer.android.com/design/index.html
[4]https://getmdl.io
[5]http://getbootstrap.com

D. Act

The final phase of the cycle comprises of reviewing the evaluation. Depending on the development process, the results might be immediately implemented or added to the backlog for the next iteration. At this point, developers can create their own lightweight *Styleguide* that contains rules and guidelines for interfaces [7, cmp. p. 165] based on the lessons learned during the current iteration. Over time, the Styleguide will aggregate more and more information that can not only be used to validate design decisions in the future but also to guide other developers as well as new employees.

V. Conclusion

In a market that is as competitive as the software market, manufacturers constantly need to find factors that give them a competitive advantage over their competitors. As studies show, improving the usability of the applications developed by the company can offer exactly that. In the consumer market, and particularly in the mobile application sector, this realization is largely general knowledge – and due to the consumerization of business software, the demand for polished and user-centered interfaces in the business sector is growing too. Despite this potential, the amount of resources dedicated to usability measures taken in the course of the development of a software is still comparatively small. Therefore, this paper described lightweight techniques that can easily be integrated into the existing software development process. As a result, they can be applied by individual developers even if the budget is limited.

Since one of the central issues of HCI is to understand the people that will use the application, the methods focus on two key aspects: Firstly, getting a better understanding of the users and secondly, evaluating the design decisions made using that knowledge. In the planning phase of the product, we as developers can use Personas and Storytelling to build empathy for the fictional characters which makes it easier to imagine how they would behave in certain situations. With this in mind, we can create simple Prototypes of the GUI. Meanwhile as well as during the actual development, we can use existing knowledge and resources such as design patterns, platform guidelines and toolkits as a solid foundation for the interface. To evaluate the interface, we need feedback from the customer. In addition to that, we can fall back on the Personas and perform a Cognitive Walkthrough of the application from their point of view. A more systematic approach is to review the user interface with regard to Usability Heuristics like visibility, consistency and error prevention. Both serve the purpose of discovering potential issues. Afterwards, we should record the insight we gained in a Styleguide so we can get back to it in future iterations and projects or give it to other developers.

In contrast to similar work, this work does not try to transform the development process itself. Instead, it suggests methods that can be added to the typical software development process. Although this provides a good starting point, it will not be enough on a long-term basis since it still revolves more around the experts creating it than the actual users. But given the positive impact that HCI has on the success of an application, it might help to establish it as a more central factor. In the end, this would benefit everyone: Since usability draws its competitive advantage from being highly focused on humans, it has not only the potential to increase the turnover of a company, but also to enrich the lives of the people using it.

References

[1] J. Nielsen. (2011, Sep.) How long do users stay on web pages? Nielsen Norman Group. [Online]. Available: https://www.nngroup.com/articles/how-long-do-users-stay-on-web-pages

[2] G. Lindgaard, G. Fernandes, C. Dudek, and J. M. Brown, “Attention web designers: You have 50 milliseconds to make a good first impression!” Human-Oriented Technology Lab, Carleton University, Ottawa, Canada, Mar. 2006.

[3] M. Woywode, A. Maedche, D. Wallach, and M. Plach, “Gebrauchstauglichkeit von anwendungssoftware als wettbewerbsfaktor fuer kleine und mittlere unternehmen (kmu),” Institut für Mittelstandsforschung, 2011.

[4] Consumerization – gartner. Gartner. [Online]. Available: http://www.gartner.com/it-glossary/consumerization

[5] D. Hering, I. Emons, and A. Ismer, “Simply usable – abschlussbericht,” Fraunhofer FIT, 2015.

[6] “Methodenhandbuch nutzerzentrierte entwicklung,” Kompetenzzentrum Usability im Mittelstand, 2015.

[7] T. Stapelkamp, *Interaction- und Interfacedesign*. Berlin: Springer-Verlag, 2010.

[8] Hewett, Baecker, Card, Carey, Gasen, Mantei, Perlman, Strong, and Verplank. Curricula for human-computer interaction. ACM SIGCHI. [Online]. Available: http://old.sigchi.org/cdg/cdg2.html#2_1

[9] J. Preece, Y. Rogers, and H. Sharp, *Interaction design: beyond human-computer interaction*. New York, NY: Wiley, 2002.

[10] C. Moser, *User Experience Design*. Berlin: SpringerVieweg, 2012.

[11] P. N. Johnson-Laird. Mental models and human reasoning. [Online]. Available: http://www.pnas.org/content/107/43/18243.full

[12] D. A. Norman, *The design of everyday things*, revised and expanded edition ed. New York: Basic Books, 2013, previously publ. as The psychology of everyday things.

[13] H. Brüggemann and P. Bremer, *Grundlagen Qualitätsmanagement*. Berlin: SpringerVieweg, 2012.

[14] W. Quesenbery and K. Brooks, *Storytelling for User Experience*. New York: Rosenfeld Media, LLC, 2010.

[15] G. Dobrescu. Week 2 – personas. [Online]. Available: https://ginadobrescu.wordpress.com/outreachy-program/week-2-personas

[16] M. Bowers, J. Cao, and M. Ellis, “The practical handbook to rapid lo-fi prototyping,” UXPin, 2015.

[17] B. Gremillion, J. Cao, and K. Zieba, “Tactical ui patterns,” UXPin, 2015.

[18] J. Nielsen, “Usability inspection methods,” J. Nielsen and R. L. Mack, Eds. New York, NY, USA: John Wiley & Sons, Inc., 1994, ch. Heuristic Evaluation, pp. 25–62. [Online]. Available: http://dl.acm.org/citation.cfm?id=189200.189209