RESEARCH AND DESIGN OF USER EXPERIENCE

Reader

Module 6 Creative Technology
Intelligent Interaction Design

Dennis Reidsma, Robby van Delden
Andrea Papenmeier

Module ID: 202400411,

13th of November 2025, v10

UNIVERSITY OF TWENTE.

Table of Contents

TABLE OF CONTENTS **2**

1. INTRODUCTION 4

PART I – GROUNDED RESEARCH AND DESIGN **5**

2. RESEARCH AND DESIGN OF USER EXPERIENCE 6
Grounded design *6*
The user in the heart of our work *6*
Design happens in iterative processes *7*
Design is supported by research *7*
Design is about making decisions… *8*
…that are supported and validated through arguments based on research. *8*
Research is structured through Research Questions *8*
Research is reported in fairly standard ways *9*
A spark jumps the gap *11*
3. WHAT IS (GOOD) RESEARCH?? 12
Novelty and originality *12*
Relevance and impact *12*
Generalization and new knowledge *12*
Methodological rigor *12*
Formative versus summative research outcomes *13*

PART II – ILLUSTRATIONS AND USER CONFRONTATIONS **14**

4. USER CONFRONTATIONS 15
Some User Confrontations should remain devoid of your own ideas *15*
We are interested in different things across the different phases of a project *15*
Literature search as a type of user confrontation *16*
Ethnographic methods Interviews and observations *16*
Observation of users with similar technology *17*
Surveys *17*
User testing with prototypes *17*
Practical organization *18*
Analysis of (interview) content: from qualitative to more quantitative *19*
Working with users in a good way *19*
More reading: TOOLBOXES *20*
5. ILLUSTRATIONS 21
What is the role of an illustration, and for whom is it meant? *21*
Scenarios and personas *21*
Prototypes *22*
6. FIRST USER REQUIREMENTS 24
Examples of requirements *25*

PART III – FROM USER NEEDS TO REQUIREMENTS TO SPECIFICATION **27**

7. FROM NEEDS TO REQUIREMENTS TO SPECIFICATION 28
Customer journey map as a communication tool *28*
Decomposition and modules *29*
Functional requirements, non-functional requirements, and constraints *32*
Communicating Requirements: formulation for testing, prioritization, interrelatedness, and contracts *34*
The end user as starting point and end point *37*
V-model as an organizing methodology: increasing detail and testing *38*
Ideal world, manufacturing, and prototyping *39*
Conclusion *40*
Further Reading and References *40*

PART IV – MAKING BEYOND RESEARCH **42**

8. IMPLEMENTATION 43
Some thoughts on more detailed design of hi-fi prototypes *43*

PART V – MEASURING SUCCESS 46

9. PRODUCT VALIDATION BY RESULTS 47
Investigating measures - efficient, effective, satisfaction, and UX. *47*
The pragmatic balance between validated and attainable questionnaires *48*
Analysis *49*

1. Introduction

This reader contains part of the materials that you should learn for the exams of the theory part of the study unit “Design and Research of User Experience” in OSIRIS, part of Module 6, Intelligent Interaction Design of the Creative Technology program. Other materials include a book, videos, and papers; the complete overview can be found in the **Course Manual** of this same course (which also explains in detail the content and organization of the group project).

Together with the other mandatory readings, this reader provides theory and background to the process of Research and Design for User Experience. It also helps you to build your toolbox of methods and techniques for user centered design and evaluation of HCI. The exam materials are enough to complete the project with a passing grade. The reader also provides pointers to additional knowledge and methods for user-centered design and research. These optional materials are mentioned in separate insets.

Using what you learn here, you will hopefully be able to confidently navigate the endless space of current and newly developed methods and techniques to yield the best and most relevant interactive product in any new situation for any new domain and user group.

Part I – Grounded Research and Design

2. Research and Design of User Experience

Intelligent interactive systems may be developed for many different reasons: to enable new experiences in daily life; to improve efficiency, performance and/or comfort in doing certain tasks; to change people's behavior in certain situations; etcetera. When you set out to develop novel products in a certain domain, your client generally won't be able to tell you what exactly you should be making. Neither do you necessarily know in advance what you are going to make. And if you think you *do* already know what to make, it may easily turn out that you should make something different – which you will find out when you get to know your product and your user better.

Grounded design

The approach that we take is one of grounded design: A good professional is able to take an *ill-posed problem*, navigate through *grounded design decisions* to a good *product solution*, and *justify in context* why this product, based on these decisions, is the right one for this particular use context. In this approach, we simultaneously design partial solutions and product concepts, construct (and tell) a story that explains the value of the product, and argue how the story and design are grounded on the underlying contextual knowledge and insights. These three activities mutually inform and strengthen each other. By thus placing our design and its underlying Design Rationale into context, we can clearly show what is unique, novel, and worthwhile about our product. This is not only good for marketing purposes. It also helps us to develop better products because we, as the designers of these products, understand better what makes our product valuable.

The *successful* development of interactive systems is then not only the result of technical and aesthetic skills in developing a product that works and that looks nice, but also the result of our ability and willingness to keep reflecting on the *conceptual side*, to keep evolving and refining our product ideas. Reflection leads to insight leads to new, grounded, design choices and modifications.

The user in the heart of our work

The *functionality* of a product – what task it should support, what actions we should be able to do with it – is obviously important. However, nowadays, designers tend to (additionally) focus on the *User Experience*. User Experience[11] (UX) can be a lens through which to view the interaction between user and product, but also to look at the user domain in itself (without the product) to describe what is important there. This helps us think about more than just practical tasks carried out with our product, including aspects such as a user's feelings, emotions, social patterns, physical and embodied aspects, values and goals, etcetera.

Our products are made *for users* to improve the life of their users. Our products are furthermore made *to be used by users* – they are artifacts of interaction technology. This is why we work by putting the user at the center of the design process and getting back to the user in every phase of our project. If we understand our users and the domain well, there is a bigger chance we will develop something that they need and like. We need to find out what goals we should strive for with our product, as well as what form the product and the interaction should take. If we present our prototypes to users early on, we will get feedback about our system's strong and weak points. We will see what the product could mean to them, but also how the interaction between user and product unfolds in reality. This increases the chance that we will catch misconceptions, wrong design decisions, or opportunities for improvement early enough to still do something about them. This is why we keep involving users in various ways. Not only to make better products, but also to make it more likely that the user will understand and appreciate our product.

---

[1] A nice introduction to what User Experience is can be found in the UX Whitepaper at www.allaboutux.org (not part of the exam!). In addition, if you have the book "The Art of Game Design" by Jesse Schell, it might be worthwhile to re-read the chapter about the Essential Experience (also not part of the exam)

Design happens in iterative processes

All of these things (user-centered design, storytelling, and evolution of your product concept) are best served with a parallel approach of simultaneous making (products and ideas), confronting (with users and stakeholders), and reflecting (on everything you do and make). These activities take place in parallel.

All three activities are relevant throughout the project; we keep going back and forth between them. For example, user confrontations are carried out in the beginning, when we are still trying to understand our user, but also towards the end, when we are more into testing and finetuning our final product. Nevertheless, one can point to recognizable *phases* in which these activities are carried out. One (well-known) definition of these phases is the multi-stage process of Design Thinking[2] originally introduced by Hasso-Plattner: Empathize, Define, Ideate, Prototype, Test.

As suggested in Figure 1, the full design process is generally understood as a spiral process. In such an iterative / spiral approach, one can keep returning to earlier phases of the project. For example, using the outcome of a user test with a prototype, one can redefine the initial goals of the product and re-enter into a new prototyping stage. But although we go back and forth between activities and phases, there is a certain convergence in the sense that as the process unfolds, we get a better picture of what we have to make, what goals we have to pursue, and what the user really thinks of it.

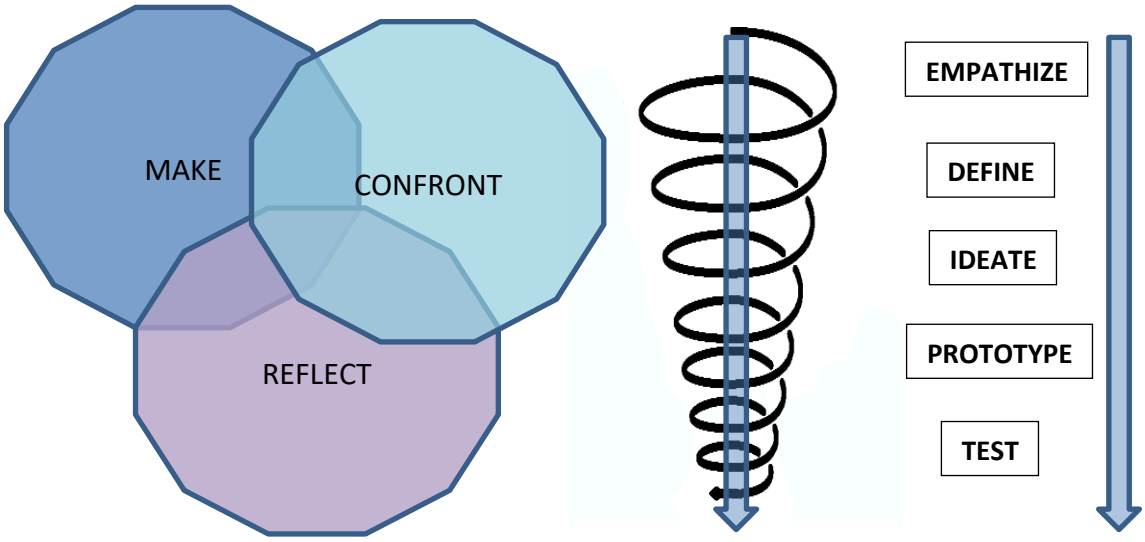


Figure 1: Activities and phases in the iterative method underlying our design work

Design is supported by research

When looking at the activities across the various phases described above, there are two lenses one could take: the lens of *design activities*, which we focus here on the decisions that the designer makes along the way, and the lens of *research activities*, which we focus here on the information and arguments that the designer collects to support and validate these decisions. Together they form the heart of our *design rationale*: the story of why we make this product, and why we make it in this way.

[2] A website with much interesting information on this can be found here: https://www.interaction-design.org/literature/topics/design-thinking

Design is about making decisions…

Design ultimately leads to a product through a series of *decisions*. Decisions about which problem we choose to tackle and what kind of product to make; decisions about what form the product, the interface, and the interactions should have; decisions about the control paradigm that we want to incorporate in the interaction; decisions about the content to include in our product; and many, many more decisions big and small.

…that are supported and validated through arguments based on research.

User research yields the information on which we ground our decisions. Sometimes, we first do the research (e.g., interviews concerning user needs and problems in the Empathize phase) and make decisions based on the results. Sometimes, we first make a decision (e.g., about possible creative solutions that we could pursue), and subsequently do research to validate our decisions and better understand the possible outcomes of our decision (e.g., how does the product change the user's behavior?). Such research will help us better support the original decision with new argumentation.

Arguments, based on information, are the grounding of your design rationale. They inform your decisions. But the making of decisions can sometimes also come first.

Design Rationale: Grounded system design decisions are combined into a cohesive story about our product. This helps us to communicate clearly about what we are making, but also helps us reflect on whether we are making the right thing.

Research is structured through Research Questions

To make decisions we need the insights. To have the insight, we need to search and research. This is driven by research questions.

Again, nothing in the process is strictly linear. The questions lead to information which leads to decisions – decisions lead to questions that lead to information – if we do not have a first inkling of what decisions are coming up, or what information might be available, it will be very hard to define our Research Questions! So the Research Questions are highly dependent on the product that we are pursuing. Yet, we can say a few words about what Research Questions we can use.

First, there are three general *types* of questions[3] that we always find in research and design of User Experience.

- A *knowledge question* is answered by collecting and analyzing information such as interviews, observations, or literature. ***Q***: What are typical activities and experiences in a museum setting? ***A***: The following activities and experiences: …
- A *design question* is answered by making something. ***Q***: What is a good algorithm for solving

[2] For those interested in reading more about this: there are many starting points, but two sources could be the paper by Mader and Eggink about a Design Process for Creative Technology, and the work of Wieringa cited in that paper. Both papers are not part of the exam materials.

X? ***A***: this one, because… ***Q***: What is a good product for persuading people to practice their physiotherapy exercises? ***A1***: the one I just made, because… ***A2***: NOT the one I just made, because…

- An *evaluation question* is answered by doing an experiment or study with your product. ***Q***: what is the technical performance of my product? ***A1***: it can run uninterrupted without breaking for X hours consecutively. ***A2***: It fails in the following conditions: … ***Q***: What is the impact of my solution on the daily practice of my users? ***A***: It changes their experience in the following ways, … ***Q***: How does our solution change the user's social experience? ***A***:…

Second, there are things to say about good and no-so-good Research Questions. A good question is answerable in a concrete way – and not with only YES or NO. The answer should provide us with new and specific input for our decisions.

Here are some examples of bad Research Questions: *"Do people like prototype* A*?"; "Is idea* B *a good idea?"; "Would idea* C *work?".* These are answered with YES or NO only; they are too unspecific, and the answer will not in any way give us new directions in which to make decisions (regardless of whether the answer is YES or NO). *"Does the user understand our interface?"* is a useful question in the sense that a NO for answer tells us to change things, but it does not yet tell us *what* we should change so it is still not very good. A question such as *"Do users prefer A or B"* leads to some more specific insights, and at least then we can make a decision about whether we proceed working on idea A or idea B. Yet, its answer still does not provide us with insight on how people exactly respond to the options A and B or on what are the actual differences between how they perceive the options. A better question would be something like, for example: *"Which of the following activities is better at triggering the user to contact family and friends proactively, and how do they actually follow up on that trigger? What type of action does the user then take, and how do these actions differ with the type of triggering activities?"* The answer to this question is an inventory of all kinds of actions and ways of responses that users exhibit after being triggered with one activity, or another – and such an inventory can be used to productively think about ways of using these to refine our product idea.

It takes a lot of experience to come up with productive Research Questions, but the examples above do give a starting point for how to think about these. Furthermore, you can use some simple checks with your team, such as when I try to answer the research question does it require me to design and build something? If the answer is no, you probably had a knowledge question of an already existing/known solution. Similarly, when you look at the entire project, do you cover questions that should be answered with literature and interviews? Do you have questions that fit low-fi prototyping with playtesting to improve your design? Do you have question that make you rethink the direction you should pursue and allow you to steer? Do you have questions that require you do an end evaluation and is it clear whether this requires an evaluation which is quantitative (number and outcome) and/or qualitative (descriptive insights)?

Research is reported in fairly standard ways

As part of documenting our design rationale, we report on the underlying research that we carried out. A really compact statement about the goals of this reporting could be the following.

First, the reader should obtain *understanding* from our writing.

- The reader should understand why we did the research – what did we want to know, and why did we think that was important?
- The reader should understand what we did and understand what are the outcome/results

- The reader should understand the implications of the results – in our case, understand how the results lead to design decisions and how the results contribute to the design rationale

Second, the reader should feel they can *trust* our reporting.

- The reader should be able to trust in our results, which is achieved by showing in enough detail what we really did so the reader can judge whether our methods were good enough
- The reader should be able to reproduce our work on the basis of what we report – among other things because that contributes to the above trust
- The reader should trust that the answers that we give are not beyond the research questions that we asked and, thus, the research that we did. We can not provide answers other than what we asked questions about and what we really investigated. Of course, sometimes we want to say additional things, share insights that we personally found while doing the research. But if it is not an answer to an actual RQ, such thoughts belong under DISCUSSION or FUTURE WORK, not in Results or Conclusion.

When reporting research, we use a fairly fixed structure[4] that contributes to the above goals. Some of these sections may be combined into one, or split in two, but generally the below sections are recognizable in all research reports.

Introduction. A brief (!) description of what the document is about.

Motivation / background. Why are you doing this/why is this important to address? What is the background of the work you are reporting? What will be the impact of this work? What is new/different?

Research question. It is quite a challenge to find good research questions for every phase in a user experience design project. The previous section discusses this. Make sure that you clearly frame your research questions. Quite often you have more than one., or a main one and several sub questions to support answering it.

Similar / related work. What have other people already done that is similar to your work?

Methods / Materials / Approach. What steps will you take to answer your question? Methods include reading literature, performing a brainstorm, making a prototype, developing an algorithm, interviewing people, designing and executing an experiment, etcetera. Your approach depends a lot on the type of question that you are answering, and hence on the phase of the project. So: explain why you apply a certain method at a certain point to answer that specific question. Sometimes, you have multiple questions that are also answered through multiple methods. For example, the question “what problems do teachers experience when they try to teach music?” can be answered through literature search, interviewing teachers, observing teachers, … From this section, the reader should be able to reproduce your work in the exact same way.

Results. What are the direct results? Here, you describe the prototype, present the numbers from your experiment, summarize the outcome of the interview, whatever constitutes your result in a particular phase of the project. Do not yet turn to *interpreting* the results, just the objective parts go here. This allows the reader to assess the implications of those results on their own, apart from your interpretation.

Discussion and conclusion. In the final part of your reports, you explain the impact of the results. What is your conclusion? What is the answer to your research questions? What is the impact on your product concept and design, based on the insights that you gathered? This is where you will present parts of your *systems design rationale*: why the systems product is as it is. Your future

[4] Also refer to the Skill sheets by Van Tulder: various sheets may be relevant here; e.g. E3, E4, E5, but also others; see the table of contents of the book (note that Van Tulder is not part of the exam materials).

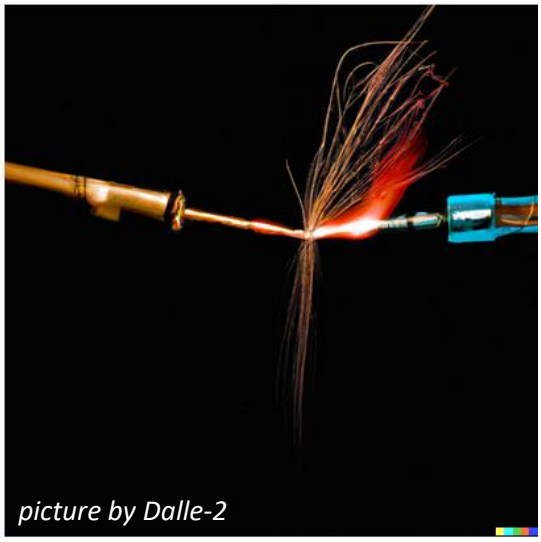
client/teacher and fellow students often need to know why a certain decision is made. NOTE the remark above about not providing *conclusions* that go beyond the Research Questions that you asked.

References. Finally, the bibliography or "references section" provides all academic references used in your report. Use a professional standard that you feel comfortable with, and use it consistently.

The above structure is not valid only for a full research project report. In most cases you can even apply it to the single chapters in which you report one phase of your project.

A spark jumps the gap

Research in itself is not entirely enough to make our decisions. There remains a gap between related work, complete insight into the literature, expert knowledge, and lists of requirements on the one hand and an experience, a task, and a useful interaction context that leads to pleasurable interactivity on the other hand. In the sections above, we described a rational process: we collect information and arguments, make decisions, and further justify them with more argumentation. However, there is one step missing. If we know everything about the context, we still don't know what "the magic thing" is to make. The arguments are needed to support the decisions, but they do not completely *determine* the decision: we need a spark to jump the gap.



picture by Dalle-2

This part, the creativity part of our work, is not covered in this reader. In previous modules, you learned about creative thinking and creative acting. But it is good to remember that creativity remains a crucial part of our work, even in systematic research where we ground design decisions in information and argumentation, and can be described to some extent by even reporting on the used ideation methodologies.

Immerse yourself in the world of the user and their daily context.

Make prototypes to explore the space of possible solutions.

Confront users with your work, often, to see what works and what doesn't.

Reflect on your findings, so as to better find out where in the space of possible solutions you need to go.

Use research methods to get more valid and reliable insights to build upon.

And keep doing this again and again.

3. What is (good) research??

Intelligent Interaction Design is, among other things, about research with users. On the one hand, user research helps us develop and improve technological products. It may be clear that some of the coming pages will be about user research, in a way, and its relation to design. On the other hand, there is *scientific research, with users*, that allows us to present our work at conferences through scientific, peer-reviewed, and hopefully read & cited papers. Clearly, not every scientific experiment leads to improved user products. Equally clearly, not every bit of user research is immediately fit for publishing as scientific work. Yet the two types of research with users are related. They are not the same, yet they are close and similar in many ways. They especially share some of the characteristics of what is considered to be "good" research with users. Here, we briefly discuss them from the perspective of Research and Design of User Experience and our lens of design rationale / grounded design decisions. In the final phase of the project, all these aspects come together as you prepare plans for an experimental research study concerning your interactive product.

Novelty and originality

A good research project concerns novel, original work. In intelligent interaction technology, this has three angles: innovation in the technology itself (new approaches and solutions), innovation in the user domain (transformation of the user domain, as also discussed in the corresponding chapter from Schell), and innovation in the interaction (new ways of interacting between user and technology). In the ideal case, these combine and reinforce each other in the same project.

Relevance and impact

Good research projects target a problem, challenge, or opportunity that has *relevance* to users (so we really care about success) and promise *user impact* if success is achieved. Research supports the design of something meaningful to users, and the outcomes of the research should lead to further grounding of our design (what do the outcomes mean for our product?).

Generalization and new knowledge

Scientific impact is achieved if the research results offer new knowledge or theory or if they generalize beyond the particulars of this research study and product. What does our research teach us that will help us in FUTURE design projects? Did we gather generalizable fundamental insights regarding users? The technology? The design process? Do we now fundamentally understand better how this kind of technology impacts users? Can we formulate *guidelines* for future design work? The more we show this kind of generalization, the wider the reach of our work becomes.

Methodological rigor

Compared to the "user tests" in the lo-fi prototyping stage, the user studies in the later phases are more extensive and more systematic: you will prepare a properly designed evaluation experiment rather than informal testing, and prepare analysis methods that are more formal using the right statistical methods. For research to be scientific, it has to be valid, reliable, and reproducible. This is also partially discussed in the earlier sections on research questions and reporting research. You need to use proper study designs and measurement methods that have been built on theory and validated in practice, and need to work with users that are properly representative of your target audience. This is also what a large part of the Lazar book is about.

For more extensive information on how to design your research study, we refer to the relevant book chapters of the Research Methods book and to the corresponding HCI lectures.

The exact measures (e.g., questionnaires that you use) depend clearly on the goals of your system. Often, it is worthwhile to have a look at www.allaboutux.org, https://hci-studies.org/, https://www.asarif.com/notes/scales.html and see what questionnaires and

observation methods are listed there. Also, you can ask teachers to help you find good measures, and we will discuss some measurement instruments in the courses of this module.

Formative versus summative research outcomes

The project's final phase is about research, and implementation (discussed in Part III – From User Needs to Requirements to Specification of this reader). Similar to the lo-fi phase, you confront your system with real users. However, where in earlier phases, your user studies will be aimed at uncovering potential improvements and making design decisions, at a later point, we will be interested in finding out how good / new / different your system *really* is.

This is related to the difference between **formative evaluation** and **summative evaluation**: The former is what we will do in the lo-fi phase, evaluating something to find out what the strengths and weaknesses are and the opportunities for improvements, as well as to provide feedback and trigger new design decisions. The latter, summative evaluation, is focused on setting a definite line: "To what extent did we make what we meant to make? Does our system satisfy certain specific criteria? Is it fit for purpose [yes|no] ? Was feature X indeed important for the success of our product?".

Introducing a new intelligent interactive system in a user context is an intervention that we want to know the effect of. We claim that the new product makes our work more efficient – can we prove this? We think our modifications to the game increase feelings of social presence – can we quantify how much? We designed the system to lead to more and better learning in a school class – can we show formally how this works out? With the design of an experimental study, we attempt to find definite answers to these questions.

What makes a product valuable?

The user, who in interaction ascribes meaning to it.

If the designer has a strong vision on what this meaning is, this may influence the user towards the same.

Then again, it might not.

We better check this – just to be sure

Part II – Illustrations and User Confrontations

4. User Confrontations

The User is in the center of our User Centered methods in multiple ways. We design —and validate our designs— with, for, and around users. They are our informant to learn more about the domain; the test person with whom we validate our design decisions; a source of inspiration for additional design ideas and decisions; and the stakeholder who, as actual user of the product, is the most affected by the outcome of our design process —the interaction and the experience.

There are a uncountable numbers of methods for involving users, depending on the phase of the project and the goal of the user confrontation. In this chapter we go through a few aspects of these user confrontations. Two main types of end users that we are concerned with are end users (those who will use our product) and domain experts. Regarding the latter, as Jesse Schell says in his chapter on Transformations, there are experts who know everything about the domain, and experts who are really good at explaining their knowledge to outsides – as the two types are not always combined in one person it certainly pays to have both types involved in your project.

Some User Confrontations should remain devoid of your own ideas

First of all, there are two broad categories of things that we need to know from users: (1) knowledge and insights about the user domain *independent of our preconceived technology/product ideas*, and (2) knowledge and insights about the users response when they are confronted with our product ideas. It is important to separate these. In early stage interviews and observations, you do *not* yet introduce your ideas for solutions, products and technologies into the conversation. Because when you do that, the remainder of the conversation will be about your ideas (so: you will focus on *sending* information that you already know) rather than about the aspects of the user domain (so: *receiving* information that you did *not* yet know). In later stage interviews and observations, in contrast, you *are* interested in the users' response to being confronted with your ideas and prototypes.

We are interested in different things across the different phases of a project

In the early stages of a project we are working on Context Analysis. We are interested in the characteristics of a user; typical activities and tasks of a user in the domain; typical situations users find themselves in; problems and issues; social interactions; desires, goals, values; current technology use and expectations thereof; possible user roles; etcetera. We collect information from many sources; we frame our problem statement and identify the current scenarios in which we see opportunity for interactive products; and we validate these in further confrontation with users. Why is it so important to gather such a broad view of the user? Since we do not yet really know what we should develop (or at least we are not certain that our idea is the right one), we need to find as much information as possible to give our choices a firm grounding in reality.

Of course, you will not limit yourself to the existing situation only: interactive technology is also great at allowing us to build something completely new. Your system may address a **latent need: something that the user did not yet know they needed until you designed it**. This may involve defining new goals, new activities, etcetera. In Ideation and Prototyping phases we may work with users to get inspiration for products and solutions. There are whole schools of design devoted to the idea that you do not only design *for* users, but can also collaborated *with them* in the design activity itself. A major challenge in that domain is that it requires the designer to have the right personal skills in guiding the user through this process.

In the Testing and Evaluation phases we (iteratively) look –with the help of users– at whether our product works out as we expected, and whether it has a beneficial effect for the user. User testing is an iterative process where we change and refine our ideas throughout numerous confrontations of our technology with users.

After a product has been fully implemented and it is in actual use with real users, we are still interested in further user confrontations: it turns out that people use our product different from what we expected or designed; this so-called “appropriation” (among other terms) is another whole domain of user research.

Literature search as a type of user confrontation

The first way to get into contact with (information regarding) users is by proxy: on the basis of literature and related work. Literature can serve as a theoretical confrontation with knowledge about users. There may be papers about the user domain in general; papers about certain typical problems; papers about approaches to solve these problems *without* technology; papers about underlying causes from which the problems and challenges originate. Furthermore, once you have identified a general direction of technology and interaction that you want to pursue, there may be papers on that technology in itself, its uses in other domains, possible interactions and user experiences associated with the technology, underlying knowledge about the type of interactions that you are looking for, etcetera..

Reading scientific and popular scientific literature is a good source of information about your target domain and user, but also about possibilities for new product ideas. You can read literature and documentation for an analysis of existing *similar technology*, or papers by people who have similar aims and motivations with their work as you have for this system, or papers that discuss characteristics of similar user groups, or papers on systems using similar technology, etcetera, etcetera. To make your analysis of related work effective, you need to make sure that you not only describe what the related work is, but also explain why you use this paper, and how it applies to your own work, what insight you take from it and how you can use that in your design.

Analysis of related work obviously starts with *finding* related work. Not just blogs, Wikipedia, or other websites, but preferably mostly scientific articles. Keep track of the keywords that you used to find materials (it may help you find even more!); if a paper seems relevant, verify its quality (looking at the date of publication, the quality of the journal or conference, number of citations etc), look at its references, find more work by the same author, or use the paper to refine your set of keywords. Such details are reported as part of your “method and approach” sections of your reports. It is also easy to find *too much* related work. Quickly judging which papers you should *not* read completely is also an important skill to develop. Some hints on reading papers can be found in the Skill Sheets book of Van Tulder, e.g. Sheets A18, C5, C7, C8.

Ethnographic methods Interviews and observations

Chapter 8 of “Research Methods” gives an overview of several types of **interview** and discusses extensively how to prepare and to carry out interviews. **Direct observations** are discussed in the video Participant Observation (see Mandatory materials), by Scott Klemmer. Klemmer discusses several interesting questions such as: what can you learn from observations, why is there a difference between what people say and what they do, and should you give users the role of designer?

Both methods are important and necessary. Interviews give the opportunity to ask people about their thoughts and to elaborate on answers given. On the other hand, an interview does not always give a complete and correct idea of how people act. People often are not entirely aware of how they act in certain situations, or find it difficult to describe concisely what they do in which order. The only way to get that information is direct observation of people in the right context. The video “Participant Observation” (see Mandatory materials) explains this clearly.

Sometimes it is also useful to survey and/or interview people who are not of the target group themselves, but know the target group very well (teachers of children, the care staff of a nursing home, etcetera).

Observation of users with similar technology

Direct observation is also a very worthwhile method if there is similar technology available. You can learn a lot about the user, the domain, and the shortcomings of existing technology by watching how people currently use technology. We are unfortunately not an all-seeing being, thus in order to see properly, it is often needed to prepare several focus points where and how to look at situations (e.g. *what do people talk about when checking out, what information exchange is there? How do peers interact in their learning task compared to with their teacher?).* When direct observation is hard, for example, because the target context occurs only rarely, you can use indirect observations. Some examples of this are pre-existing video recordings, diary studies, experience sampling, involving lead users, and involving extreme users.

Surveys

Surveys are a good way to reach many people, and also to reach people that you do not know personally or cannot meet logistically. The obvious drawback to surveys is that you cannot elaborate on an answer and ask additional questions about a topic. Therefore surveys are harder to use in situations in which you do not (yet) quite know what exactly you should be asking. They are a better fit for later phases. Interviews give you more flexibility to respond to the situation and adapt your line of questions to the information that your informant gives. Chapter 5 of "Research Methods" gives more information about designing surveys.

User testing with prototypes

In lo-fi prototyping and user evaluation we will attempt to make our product ideas more and more specific. Now, we do not talk to the user just to get to know them, but we confront them with our ideas, prototypes, sketches, etcetera, in an attempt to get a reality check on our ideas. Exactly what functionality and/or experience do we attempt to offer to the user? How will we achieve this? We will work towards a more complete picture of the product and the new situation that we will create. This picture is built through *grounded system design decisions*: you make choices founded on specific rationale and insights, iteratively, alternating between user confrontations and prototype design.

We still do not know what exactly we will make in all its specific details. We do have some first ideas. We need to choose between them: based on what is this choice made? We have some assumptions about what the user will like and how they will respond to certain aspects of our envisioned system: can we check these assumptions? We foresee partial solutions to some problems: how can we predict which of them is right?

Finding the answer to these questions requires trying out things with real users. They never respond quite like you expected. You are often forced to rethink your choices and revise your plans. Actual implementations of interaction technology are too expensive (both in money and time spent) to do enough user confrontations. That is why this phase is done with discardable lo-fi prototypes for the various questions. Making a prototype discardable also helps in making it more likely to be open for changes ideas.

A test with a lo-fi prototype is carried out to answer a question. There are two general classes of research questions that we ask at this point:

1) Can we further detail our idea of what kind of system we should develop? → The answer to such a question is never simply YES or NO (i.e., whether we have the right idea), but constitutes an increase of detail in the concept and justification that it is a good concept, or a change to another concept and a reason why the original one was not a good concept.

2) More specifically, what form should the system take? → To answer this question various versions of the systems are compared to each other: is version 1 for ... a better option than version 2? But there are also other options. For instance, functionality 1 works for users; functionality 2 does not work; variation 3 is better than 4 on a certain aspect; parameter *x* must be 25.0; etcetera. In some situations, we also use less scientifically structured, more open formative tests that can help us gain a broader set of directions and suggestions (e.g. the playtesting in the Interactive Media module where unlike proper research we change a system between tries if we are `convinced' that it should be fixed); the value of such a formative approach is in the richness and detail of the extra insights we gather in a short term, the downside is we are unable to be sure about it.

We ask these questions on different levels of abstraction. For example: Does gamification add anything useful to my system? Should we interact via touch or via free air gestures? Which interaction mechanism leads to more efficient interaction? All of these are questions that you can answer through a user test with a discardable prototype, as we will also explain in the rest of this chapter.

A good lo-fi prototype plus user confrontation leaves room for *failure*: we often learn the most when our prototypes and ideas fail in an interesting way – much more than when we learn "yes we seem to be going in the right direction". **Get *new* information out of a user confrontation with a prototype that helps the design of the system, not just to confirm the rightness of your ideas.** Specifically, the question "Does the user like the prototype?" is **almost never** a productive research question.

Again, this phase is done in an iterative approach of making (products and ideas), confronting (with users and stakeholders), and reflecting (on everything you do and make), leading to grounded design decisions. Keep it simple: to answer a larger variety of questions, use a larger variety of prototypes to keep things easier to oversee, and, most importantly, kill your darlings!

Practical organization

We evaluate our prototypes with users. We observe and ask questions about their response. Based on our conclusions, we will modify our design. Based on videos from Scott Klemmer and Jesse Schell we can better understand how to run the evaluations of these lo-fi prototypes so that you get the most insight from them. Some important aspects:

- Know beforehand what you want to learn from the evaluation (as Schell says, each test aims to answer a specific question).
- If you want to compare multiple alternatives, decide whether you want to work with the same users for every alternative (within subjects) or give every user just one option (between subjects, see Research Methods section 3.3.1 of the book).
- Determine beforehand as concretely as possible which steps you want to lead the user through and which task they have to complete.
- Figure out clearly which instructions you will give the user. Do you give them a specific series of steps to go through? Or just give them a goal and leave them on their own to figure out how to reach it?
- Think in advance how to get the most information from the sessions. Use observations, "think aloud protocols" (Section 10.4.6 of Research Methods) and/or use questionnaires and interviews after the session. Be sure *how to* and then actually *do* record and report those in a fitting way, separately from the conclusions but as a rationale for the choices they lead to.
- Write down clearly who of the project team will do what during the experiment

- Work with representative users that stand for the intended target user group, and in your reporting, write down an anonymized way to describe your participants with a number of participants and demographics (and, if needed, psychographics).
- Run a quick pilot with a team member as a user before doing the "real" session, to find out about wrinkles in the evaluation setup

It is important to document your prototyping work – with notes, photos, videos, etcetera. If you don't keep notes and images of your prototypes, it may be hard to report them afterwards in your final report. As long as you document your process and results well, you will end up with a collection of decisions that are well supported and justified, as well as a strongly improved (or: completely revised?) version of your story and product concept.

Analysis of (interview) content: from qualitative to more quantitative

Interviews and observations are done in various different phases of a user-centered design project, with and without prototypes. They yield a lot of interesting data, often in large amounts. The next step is to get the **relevant insights** from this data, which means you have to analyze the data and document the results.

In this module you learn some methods for analyzing and presenting quantitative results from surveys etcetera. However, interviews and observations rarely yield simple numerical results, but rather a collection of remarks made and topics discussed, with every respondent yielding different types of content. So in Data Analysis for CreaTe you will learn about qualitative data analysis. There is a very clear paper by Braun and Clarke (2006) that explains the process of thematic analysis step by step[5]. Chapter 11 of Research Methods also discusses similar approaches in their *content analysis* and *emergent coding* descriptions. There are thus various methods to turn subjective and unstructured interview and observation data into more structured and objective, systematically categorized information. The basic form of it is describing what someone has said abstracting from the exact description, then "count how many times the interview participant said something about X, or said something that means Y" – but the non-trivial trick here is to determine which X's and Y's are the relevant categories to search for in your data, as well as defining very precisely under what conditions something that the participant says will be deemed X or Y. And even worse, there is no clear guideline on how often someone should say something in relation to Y in order to still see it as important, as sometimes even with one mention Y becomes noteworthy and for other data even if it is mentioned several times still does not warrant mentioning (or creating a) Y.

Working with users in a good way

As part of the project, you will, at some point, interview end users and domain experts, and you will carry out evaluations with end users. Depending on your project, your course teachers might assist you in getting access to end users, or you might have to find additional users yourself. Teachers may put certain conditions on your working with these end users. If that is the case, you **must** comply with these conditions. If finding end users is a problem, you should ask advice from your project coordinator on getting into contact with real end users as early as possible in the project.

It is important that you understand the different roles that end users and domain experts have compared to your teachers and project coach. The end users will almost certainly know nothing about the organization of the module project, and they may not know much about the design and development of interactive technology either. It is up to you to make sure that the communication with the end users is clear, helpful, and effective. Remember: you are often asking them a favor, they are not being paid for their input, and have busy schedules; thus, giving you input will often take away from their spare time, and in the end, it is very rare that they will benefit directly from the

[5] Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, *3*(2), 77-101. Cited 178000+ times according to Google, not part of the ResDesUX Theory course exam material.

fruits of your labor. ***Successfully dealing with end users and domain experts is important.***

Some rules of thumb to take into account, at the very least, are:

- Contact people well in advance to respect their schedules
- Let them know who you are, what your goal is, and that you are doing a project for a specific class
- Give them the feeling that they are important and helpful
- Let them know that you are willing to respect confidentiality
- Ask permission before recording anything; make sure you are aware of the GDPR laws and procedures for dealing with personally identifiable data
- Thank them afterward for participating, and if they're interested let them know something back about the outcome of your project (the most engaged users and experts could also be invited for a final presentation of the outcomes, e.g. at a demo market)

Recruitment of end users is a special concern: you must plan ahead how you think to find them and approach them. Also, you must be prepared for the *ethical* side of working with real users: You need consent forms (Research Methods section 14.2), and possibly approval from the ethical committee of the EEMCS faculty. When dealing with children or with people not being able to give consent (e.g., people with Alzheimer's), be extra careful and be sure to ask advice from your supervisors. in principle, for this project, it will be out of scope due to the additional ethical considerations it will require.

If you are working on applications close to the medical domain, you must ask for specific advice concerning this – it is outside of the scope of this reader to explain the very specific ethics and rules of medical research, but note at least that it involves a few important laws; not following ethical rules (of medical research) can, in fact, lead to having a criminal record …

More reading: TOOLBOXES

This chapter covered only a very limited section of the huge universe of methods for User-Centered Design. In fact, exactly because of this fact, and because new methods for specific circumstances, target user groups, or design goals are still regularly developed, designers often depend on "tool boxes" that provide an overview of many methods that could be used. In every specific situation, then, a designer can take from the toolbox the method that seems most sensible to apply in that situation. Some examples of such toolboxes are ucdtoolbox.com, the include toolbox by EagleScience, the toolkit by Digital Society School you use for the project, and the user research toolbox by IMEC.

5. Illustrations

As our design process progresses, we collect *insights in the user domain* and we gather and develop *concepts and ideas for our product*. Both need to be illustrated during their evolution throughout the project. An illustration can be anything that helps you communicate your work to an outsider. Scenarios to illustrate domain and/or product concept; storyboards to illustrate the scenario; sketches to illustrate the looks of the product; a photo to illustrate the test setup; a flow chart to illustrate the interaction between user and system; a block diagram to illustrate the architecture of your system – a good professional has a large toolbox of illustration methods to communicate their work to outsiders!

What is the role of an illustration, and for whom is it meant?

We use illustrations to communicate our ideas to others. Illustrations show our understanding of the domain, the user, and their context in a way the audience can relate to. Illustrations show our view of the future: what is the core mission of our project (what do we really want to achieve for the end users' daily life?), where do we see the system boundaries, and how do we think the world will change around our system? Finally, illustrations show more concretely how (interaction with) our system will work and what it will look like.

Illustrations take many forms. It can, for example, be text and explanations, images and photos, mood boards, stories, videos, or physical objects: your imagination is the limit as long as it communicates the ideas you want to bring across.

On the one hand, a good illustration *sends* information: It is meant for clients and end users to give them a better idea of what we promise to them, but also for ourselves and our project team members, for engineers and other designers who need to make our vision become real.

On the other hand, a good illustration triggers the team of creators and, even more importantly, the audience to *respond* to it with new and more detailed information. On the basis of a good illustration of the user context, end users and experts can tell us in detail where and how we went wrong and what is still missing or what is most meaningful in that context. Based on a clear illustration of our product concept, engineers tell us whether it is feasible and what else is possible, and end users can give an impression of whether they expect that product to add value to their lives.

Good illustrations elicit rich responses from our audience and, as such, give us something to listen for, to enrich our own ideas by incorporating these responses. We must train ourselves to *listen* to these responses, as they are a major source of understanding and a major source of the range of possibilities to improve our ideas. Furthermore, prototypes illustrate a meaningful aspect of the user experience to the point that people can actually try it out, and we can observe their actions within that experience for further input. In the remainder of this chapter, we discuss a few specific types of "illustration".

Scenarios and personas

Scenarios are stories about the way that people act and why they do this. There are many types of scenarios, illustrating, for example, the general context, possible problems that we might want to tackle, ideas for a solution, or possible stories of future use of our product and its transformation of the users' daily lives. In the context analysis, we used scenarios to describe the situation as it currently exists. In later phases, we use them to show how people will use certain technologies in specific situations: a first description of how people will use our envisioned system. In scenarios, the focus is not on the details of the interface and not so much on the specific technological implementations but more on how the system supports certain activities or user experiences. We use the collected data to tell stories about the experiences of a person in the targeted context. Scenarios play several roles. They make design ideas more concrete and show the consequences of

the system for its users (e.g., new ways to carry out activities). They are, therefore, very useful for discussions within the design team. Scenarios are also helpful for communication with other stakeholders and form the basis for the conceptual design: storyboards, prototypes, and the setup of user evaluations can all be based on scenarios.[6]

Scenarios can be reported in various forms besides writing them out in text. A storyboard (series of sketches or images, comic-like but annotated with additional information) can display the steps in the interaction with the system or steps in an activity or challenge in the daily context of the user. A good storyboard of user interaction sketches how the system looks like in the different steps of the scenario, sketches what the user is supposed to do, and what the effect will be.[7] On the other hand, a storyboard can also have great detail in a documentary-like movie representing the scenario.

Prototypes

Prototypes are a special kind of illustration in the sense that they allow users to actually experience (parts of) the interaction with a future product. Prototypes, more than some of the other illustrations, also force the design team to be more specific and concrete about their design ideas, as they have actually to give concrete form to parts of them. Prototypes are made to explore design options and make well-grounded design decisions[8]. Prototypes are used to explain ideas to users, and to let them actively try your out ideas. Prototypes are there to be confronted with users and to get feedback on your design ideas. Often, these prototypes are meant to explore just *one aspect* of the system (see the playtesting excerpt by Schell). At other times, the prototype is meant to simulate the *whole interaction* with the user.

While prototyping, keep in mind the goal: not to make the "first version of your final system" but to explore the design space by trying out multiple variants. One of the most central principles of design is to generate alternatives. Linus Pauling, winner of the Nobel Prize, said: *the best way to get good ideas is to get lots of ideas*. The lo-fi design phase is a good moment to explore multiple design options. You have not yet invested a lot of effort in one specific design, so it hurts less to throw away certain ideas and choose other options. The crucial factor here is that the prototypes are simple, and easy to make and discard. While evaluating, you may have to change the prototype on the spot. Sometimes, the prototype depends on system behavior that you cannot yet generate automatically – for this; we use Wizard of Oz paradigms (i.e., a prototype manipulated / remote-controlled by the researcher to fake system responses).

Prototypes can have a multitude of forms. Lo-fi prototypes such as pen-and-paper prototypes are a low-cost way to test parts of the interaction with your product. You can sketch the basic layout of the various screens (what information is presented in different stages of the interaction?). By making them clickable interactive (e.g., through wireframe tools such as Balsamiq and Axure or simply by making a fake clickable screen in PowerPoint or Figma), you can also present how the user navigates through the interaction and how the different parts of the product hang together in an interaction pattern. Obviously, this works better for GUI-based products than for tangible interfaces, physical computing products, or interactive spaces. For such products, you have to be a bit more creative to come up with low-cost, lo-fi prototypes that allow the user to get part of the underlying intended UX.

---

[6] A good discussion of the usefulness of scenarios and of the different ways in which they can be used is given in the paper Scenario based design by Rosson and Carrol (2002) (not part of the exam). Another source is the paper by van der Bijl-Brouwer, M., & van der Voort, M. C. (2013). Exploring future use: scenario based design. Advanced design methods for successful innovation-recent methods from design research and design [link].

[7] See video Storyboards, Paper prototypes and Mockups, by Scott Klemmer, Mandatory materials

[8] Prototyping, and testing with discardable prototypes, is a crucial part of any design process. A good starting point for understanding this is the video "Power of prototyping" by Scott Klemmer and or the and or the sections on "playtesting" in the book of Jesse Schell (both part of the exam materials, see table in Course Manual).

Video prototypes are a way to illustrate a user experience where the input/response patterns are too complex to fake with a pen-and-paper prototype or a bit of prop-enhanced theatre. For a video to be a prototype rather than "just" a scenario illustration, it needs to give a clear impression of the form of the interaction. What does the user do, how is that turned into system input, what is the system reaction, and what does that mean for the user?

Finally, Hi-fi prototypes are closer to the real product in their form and the full experience they deliver. At this point, you have a fairly clear idea of the system you want to develop, including many of its details. In contrast to the lo-fi prototypes made in earlier phases, we now aim for a working interactive system. Design decisions additionally concern detailed aspects such as visual design (colors, layout, typography, etc.), the actual content, sounds, and images used, the precise rules of interaction or the messages of the dialog exchange between the user and system, etcetera. This is still a prototype, albeit an interactive, functional one. That is, you don't want to design "the perfect system"[9]. Also, sometimes, the product depends on bits of technology that have not yet been fully developed. In that case, parts of the interaction might still be fake, for example, because the interaction is controlled remotely by a so-called Wizard-of-Oz rather than by an intelligent module inside the system. Hi-fi prototypes help you show whether your concept delivers exactly the core experience you aim for. You may make opportunistic implementation decisions, ensuring that you deliver a system that contains exactly what you want to evaluate.

Again, we emphasize that prototypes are not meant to make the "early versions of your technical system" but to explore the design space and offer the user illustrative interactive experiences. ***These prototypes are there to be thrown away and discarded after they have served their goal (which is to make design decisions on various aspects of the system).*** From an engineering perspective, we can even say, "If the prototype doesn't break, you didn't test it". Often, it is needed to find the limits or limitations so we can address these in the next iterations.

[9] It is anyhow extremely unlikely that a consumer version of your product would ever look the same.

6. First User Requirements

Requirements are a special type of “illustration of your ideas”. They are typically more "contract-oriented" and used to communicate between stakeholders the joint understanding of agreements regarding what will be made. Often, requirements are formally agreed upon and used to determine whether the developing partner has fulfilled their obligations as they deliver their final product.

Requirements range from broad and global to very detailed. Good requirements combine these in a clear story. In our project, like any, we will return to requirements at several points in time. First, we draw and describe the users’ perspective: what do they want? Then, we work towards drafting product requirements. What should the product offer? Finally, we work towards requirements in terms of specification. To what extent should the product do this, and how can we make the engineers make what is needed without prescribing how they should do this? More of the latter we will discuss in Part V later in the project.

First, you clearly describe your goal in making this system. What do you want to achieve? What is the **essential user experience** that you want the user to have? At this point, you describe the intended system in a non-technical way (“what does it do and why”, instead of “how does it work”). This description should be compact.

For example: “We want to make a system in which people are playing a game together via internet, which causes them to feel closer to each other because they can feel each other’s touch over the internet while they play the game” (this was the original motivation for the system described in “The Art of Tug of War”).

A descriptive product concept in itself is not enough; you need to extend it with more formal Global Requirements. These serve several functions:

- Communicate with your client
- Communicate within the team (when you divide the tasks, it is important that everyone has exactly the same idea about what you will make.)
- Set the specific criteria for evaluation and success: what should you evaluate, and how will you know whether you made the system that you originally promised/aimed for?

Of course, requirements can and often *should* change during later phases, but only if all involved stakeholders, from the team and clients, agree. Paraphrasing the Agile Manifesto[10] (for developers): In the end, satisfying the customer is the highest priority. Therefore, although requirements sometimes start with contract negotiation, customer collaboration is also needed, even late in development, which may lead to changing requirements and accompanying budgets.

Functional requirements describe the practical functions that a product is supposed to offer. Most systems have the further requirement that they are effective, efficient, easy to learn, etcetera, in one word: usable. Finally, a large part of a product is not only the tasks that the user carries out with it, but also the user experience that it offers (in fact, sometimes the user experience is one of the system's primary goals). To determine if developers met their obligations (i.e., a main goal of requirements), describing the usability requirements goes beyond “usable” and needs further specifications on how often, how many, how quickly, what measurable feelings it should elicit, etc.

In other words, a designer can only translate requirements into design decisions and validations if they are ***concrete*** and ***operational***. Putting concrete and operational definitions to the requirements is crucial for user experience goals, but this is much harder due to the nature of user experience. Something that helps is to use clearly defined terms related to outcome measures based on widely cited literature and verified surveys; with this, one can get as close as possible to formally agreed-

[10] https://agilemanifesto.org/principles.html

upon obligations. However, this remains one of the challenging aspects of this design for user experience.

The formal, agreement-like nature of requirements leads to them often being presented in the form of a checklist of MoSCoW-priority ordered items of things that the system Must, Should, Could, or Won't offer. However, they should be presented within **a surrounding narrative** that puts them in relation with each other and with the overall aims of the product, in which they are organized as high-level requirements and lower-level sub-requirements. Requirements, when well presented, tell a story to the reader about not just *what* you will be making but also *why*. It should be clear how you came to these requirements: describing that well is part of your *design rationale*. This also allows stakeholders to assess the value of the requirements and helps prevent us from overlooking some requirements. The next section shows an example of requirements where the "bullet point" style descriptions are interleaved with narrative explanations.

Examples of requirements

This section presents two examples of requirements; they illustrate how you can formulate requirements for your project.

Example: hotel check out system

A hotel chain wants to make checking out easier and faster. The management knows from experience that in the morning, many people want to check out at the same time. This leads to delays and irritation for the hurried guests. One of the design ideas is a checkout station that can be placed in the lobby. Known guests can check out here without the intervention of staff.

After analysis and discussion with stakeholders, this leads to the following requirements:

Functional requirements

1. Regular guests whose identity is known must be able to checkout using the system
2. The system must make it possible to pay
3. The system must print the bill and receipt on request, or send it via email
4. The system must be able to collect the room key

User requirements

5. Checking out must take at most 60 seconds.
6. When 10 guests are waiting, the waiting time must be at most 2 minutes.
7. The new system must not take more time for hotel staff than the old (manual) method
8. The first experience must be so pleasant that at least 70% of the returning customers would choose to check-out with the system the next time.

As you can see, it is easy to relate these requirements to their underlying rationale, based on the analysis that was done before. Also, the requirements have been formulated in a concrete and measurable way. This makes it easy to see whether your system fulfills the requirements; it makes it easier to translate them into design decisions; also, it facilitates clear and unambiguous communication with team members and clients.

Furthermore, it has to be noted here that the user requirements all concern efficiency and are, therefore, in the domain of classical *usability* rather than *user experience*. Given the application, this is not so unexpected.

Example: robots for learning

A design team wants to make a robot that works with children on learning tasks. The domain is *inquiry learning* (children learning by simulation, experiment, and exploration). The underlying idea

of the system is that children are more motivated and self-directed when they work together with each other (without a teacher). This is good, but they also make more mistakes when the teacher is not there to correct them. A robot acting as a *peer learner* can tap into the first, and yet (because it is programmed by teachers) still keep an eye on the correctness of the children's solutions.

Functional requirements 1: feedback

1. The robot must be able to give feedback to the children when they are making mistakes that they cannot solve themselves.

To achieve this, the system must have the following capabilities:

1a. The robot must be able to follow the children's progress through the learning task.

1b.The robot must be able to detect some types of mistakes.

1c. The robot must be able to formulate relevant and adequate feedback to correct mistakes.

As you can see, the robot needs to be able to understand the progress of the children and detect certain types of mistakes. Of course, not every learning task lends itself well to this. So after formulating these requirements, you probably want to do further analysis into which learning domain is most suitable for this challenge – leading to new functional requirements for the system. These kinds of considerations are part of your design rationale.

User requirements

1. The child should perceive the robot as a *peer*, that is, as another "child" that is doing the same task.
2. The child should perceive the robot as having a *collaborative attitude*, that is, as if they learn together rather than compete with each other.

As you can see, these user experience requirements are still pretty vague. The challenge here is firstly to define social, collaborative, and competitive in a way that we can *measure* it, and secondly to translate it to more specific descriptions that we can use in our *design decisions.*

More tips for requirements

For those looking for jobs also outside HCI, it is good to know the practice of requirements can be daunting and legalistic, especially when designing, for instance, for the aerospace industry, more in part V. An old but fairly interesting text from this perspective is this symposium contribution by Ivy Hooks (1993) https://reqexperts.com/wp-content/uploads/2015/07/writing_good_requirements.htm. A few pointers are mentioned above, but by indicating examples, we even see more of its importance. One, writing requirements too broadly, including wording such as Etc. *but not limited to ; and/or* can also cost the developing side money, as it includes promises rather than excludes these parts, nor puts the parts out of the scope of the assignment/contract. Two, requirements do not describe ~~**HOW**~~ but rather ***What*** is needed, describing to what extent and with a clear link to why. We can easily jump to conclusions or stick to sub-optimal solutions when describing too much from the current state. Third, requirements are often too stringent; systems can fail if they do not make a fair judgment on the capabilities, and the price will often be too high. Fourth, prevent ambiguous terms. To this end ,she nicely describes good requirements are **necessary, verifiable**, and **attainable**. So, for each requirement, ask yourself "What would be the worst thing without this, and can we live with that?", "How can we measure this requirement?" and "Can we fulfill this requirement with our team, with the current state of technology, with the available money, and within the time available?

Part III – From User Needs to Requirements to Specification

7.From Needs to Requirements to Specification

In Chapter 4, we introduced User Requirements in global terms. However, an electrical engineer will have a hard time creating the robot introduced in Chapter 4 based on the requirement a *"child perceives the robot as a peer, that is, as another "child" that is doing the same task"*. As designers, we need to communicate this global user experience goal and further specify what would make the robot achieve this with accompanying rationale. At the same time, as indicated in Chapter 4, we need to provide as much lenience as possible to the engineering team to define only ***what*** and ***not how***. In this journey, we need to structure the requirements further and show how they evolve through different phases. Not in changes, as this can confuse the reader, but by adding another level of detail in each phase. For this, we build on the business-oriented book on requirements by Robertson & Robertson (2012) and the experiences of your teachers and their network with Dutch companies and institutions of various sizes. In Chapter 4, we explained that requirements need to be related to each other **within a bigger narrative** with **accompanying rationale**, and here we elaborate on how they can be communicated in a well-structured manner and with a "style such that the people receiving the information actually read it, understand it, and make use of it." (p353).

Customer journey map as a communication tool

In domains such as website and service design, a **customer journey map** shows a chronological overview from the user's perspective of how a user engages with the product. This can help to align the functionalities of each phase over time. In the Airbnb example, see Figure 9.1, they allegedly have printed the customer journey level on the wall in their main office. This journey goes from signing up in the top left, all the way to retention elements such as receiving rewards. For each stage, we can define requirements and specifications for different layers of the whole system. In the Airbnb example, they first explain the mobile app product in wireframes and mockups. Then, both the policy and service elements are summarized with keywords for overview (e.g. on a policy level: prevent violent content). These are then linked to written out properly worked out requirements, i.e., a well-structured, unambiguous, consistent description that adheres to the quality criteria of good requirements.

We will first quickly indicate how typical steps can help to create a customer journey map, for which we build on a recent but informal blog post by Nandlal. We start with **building personas,** which require a good understanding of our user (groups) through the global description provided by institutions such as Statics Netherlands (i.e., CBS) and specific understanding through observations and interviews. Then, we **identify touchpoints**, the possible types of interactions over time (commercials, manual, website, app, customer support etc). Map the **customer actions**, indicating not only the actions the user takes but also the accompanying thoughts. Following these thoughts, one can also indicate the accompanying emotions by identifying moments "of delight, frustration, confusion, and joy." This also helps to **analyze pain points**, which might also be found through the use of interviews, focus groups, and other forms of feedback from end users. The last step in the customer journey map can include **recommended solutions**. To keep an overview, see Figure 9.1. Note that the last two steps might be integrated as alternative routes or product drawings. Similar approaches can be used in game design where levels might be drawn, with below certain levels of interests, aimed for experiences, and certain skills that might be learned or virtual skills that might be gained over time to provide a feeling of progress.

The **pain points step** is related to an important lesson from scenario-based design[6] which we encountered earlier in this reader and know from the industrial design field. One relevant warning is to prevent ***'rosy stories'*****; these** are stories where the designers assume each step will go as they expect the user and technology to behave. Instead, it is worthwhile to pay enough attention to things that can go wrong in order to circumvent these and explain why. In the policy layer of the Airbnb example, we encounter restrictions about sexual and violent content.

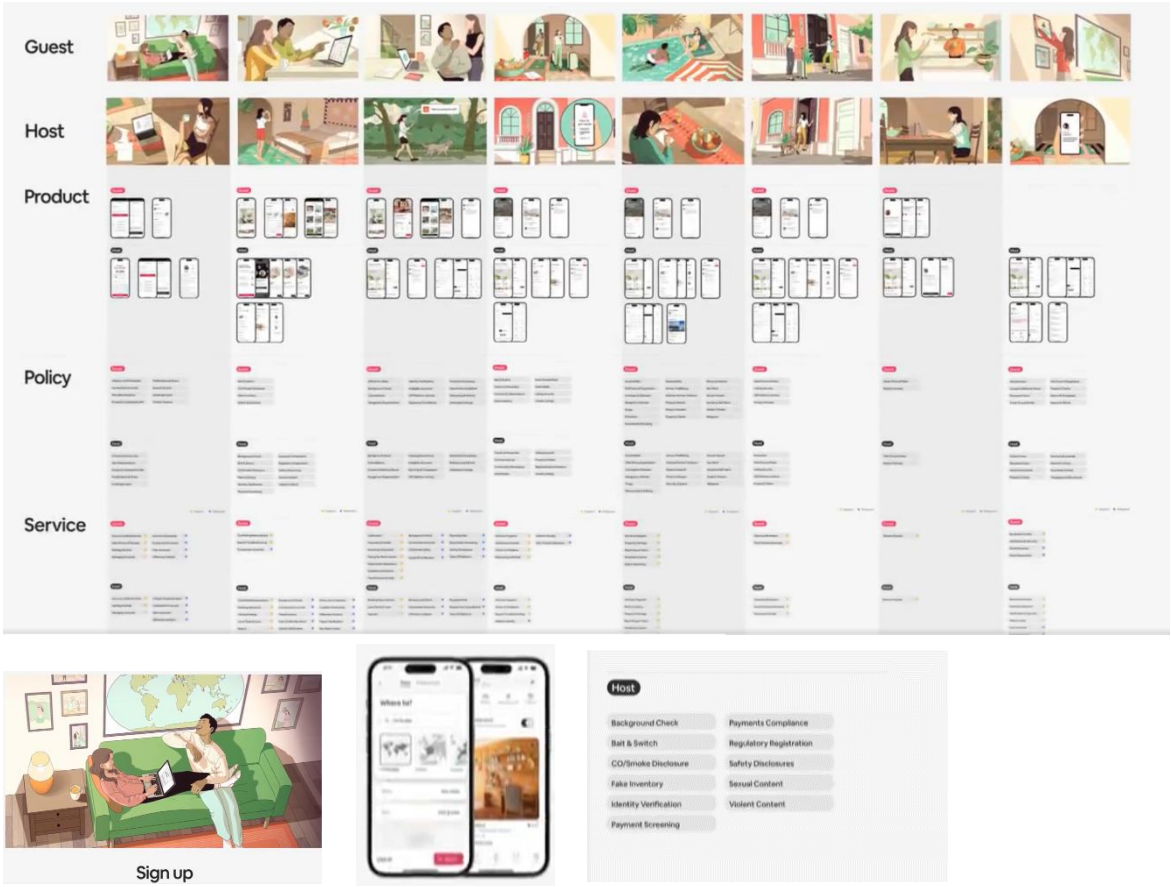


Figure 9.1 A customer journey map supposedly used by Airbnb (see video on X), on the bottom close ups from the first step, a mockup of the product when searching for a destination, a set of policies for the host (e.g. regarding sexual or violent content or for the guest on safety disclosures).

Decomposition and modules

Systems such as Airbnb have grown into rather complicated systems. Even though there are only eight/sixteen steps in the customer journey, we see hundreds of requirements on the product and service level, with several sub-systems or modules and different people assigned to these different modules. One employee might work on the sign-up part while another developer works simultaneously on the payment. Even though these tasks are rather far apart in the customer journey, these developers eventually **need to know from each other what data is required** from one part to know whether to implement in the other. In our Airbnb example, the user will sign up and has to go through certain verification steps before being able to book a house. However, before the implementation starts, one must know whether this verification is part of the sign-up module or the payment process module. Verification could be done through a driver's license, which needs to be (manually) checked before the sign-up is officially completed. This requires quite some steps to implement well, including serious privacy protection, and the payment should only be possible after this verification has been completed. Hence, the signup and verification module has to provide information to the payment module.

There are various ways in which a system can be split up into these smaller parts and to document what data is saved where and called up elsewhere. To further explain the process of **decomposition** of how a bigger system can be split into smaller parts, rather than going into database models of Airbnb, we will consider Manouk Hillebrand's example from a Creative Technology best GP Award 2021 project on communicating brain stem cancer to a child patient. The interesting part is that it combines digital and physical elements, both to record a bit of data, as well as to provide different types of feedback, covering both the smart tech and interactive media parts.

The Badge buddy system plays positive feedback sounds. It indicates certain state changes on a physical base station, shows videos about certain medical procedures, and keeps track of gone through procedures physically through badges. To know which procedure should be explained or rewarded, small badges are handed out by the medical expert. Furthermore, once a child finished a procedure, the system provides positive feedback to empower the child. Part of the system is depicted in Figure 9.2.

Figure 9.2, On the left the Badge Buddy base station with a RFID scanner, a QR code for explanation of the system, including lights, and a "toy detector", on the right a close up of the outfit to connect badges to, and below the badges with a QR code on one side and graphical depiction on the other side. In the report an interaction scenario was used like the customer journey.

The above descriptions only provided information about **what,** and not yet **how** this is done. For instance, as part of the developers' design decision on how the badges can be scanned on the base station through RFID tags as well as on a Smartphone through QR codes. The physicality of the base station adds to the interaction, linking it to the stuffed toy, and on the smartphone, it is easier to show a video based on a certain QR code. The badges can (after scanning) also be collected by attaching them to their stuffed toy using small magnets on a generic outfit fitting many stuffed toys. The latter makes it possible to attach the finished badges to a child's own stuffed toy and take it home.

Next to the customer journey, or (interaction) scenario, as part of ***how*** feedback should be given, we could make a time ***sequence diagram*** of actions performed by the user and by the system, or, in other words, the sequence of what inputs are used to trigger what kind of output by which part of the system over time. See Figure 9.3 for an example from our use case.

First, we might represent the system as a whole, taking Badge Buddy and the Smartphone together to provide an overview of the core functions (see Figure 9.4). This intermediate step can also help to assign functions to the base station and smartphone subsystems, but the pitfall is that one might consider the system from the system's point of view rather than the users'. In the lectures, we might have covered Don Norman's example, who explained that for a microwave with a digit-based interface, it might take more time to heat something in 21 seconds than 22, as you need to press or rotate to 1 rather than again using 2. This is something that we easily forget when thinking from the system functions but easier to realize when thinking from detailed user actions. This is one reason why sometimes combining partially overlapping system representations is good.

For example, in our time sequence diagram, we consider which functions go to which sub-system. For that, we implicitly make the following considerations: 1) where should the video be played? 2) where should the spoken rewards after receiving a badge be played? 3) which part should keep track of the badges? To answer the latter, we might consider that a smartphone might not be owned by the child, whereas a stuffed toy is. Thus, the medical expert might be the one handing out the badge, and then

these can be physically placed on the stuffed toy rather than virtually recording this in the smartphone; thus, instead, the physical system keeps track of which procedures have been finished. This could also help to depict the third subsystem, Stuffed Toy, separately from the base station, and a fourth human sub-system, Medical Expert, that hands out the badge in the time sequence diagram. Similarly, we might have also made different choices dividing the function differently; we could have integrated a screen into the badge buddy. This screen might automatically play the videos once the badge is scanned on the base station, preventing the need for implementing QR scanning. This would make the system more expensive but perhaps easier to use. These are trade-offs and provide a rationale for why certain choices are made as part of the specification steps related to decomposition and are typically discussed in meetings with stakeholders.

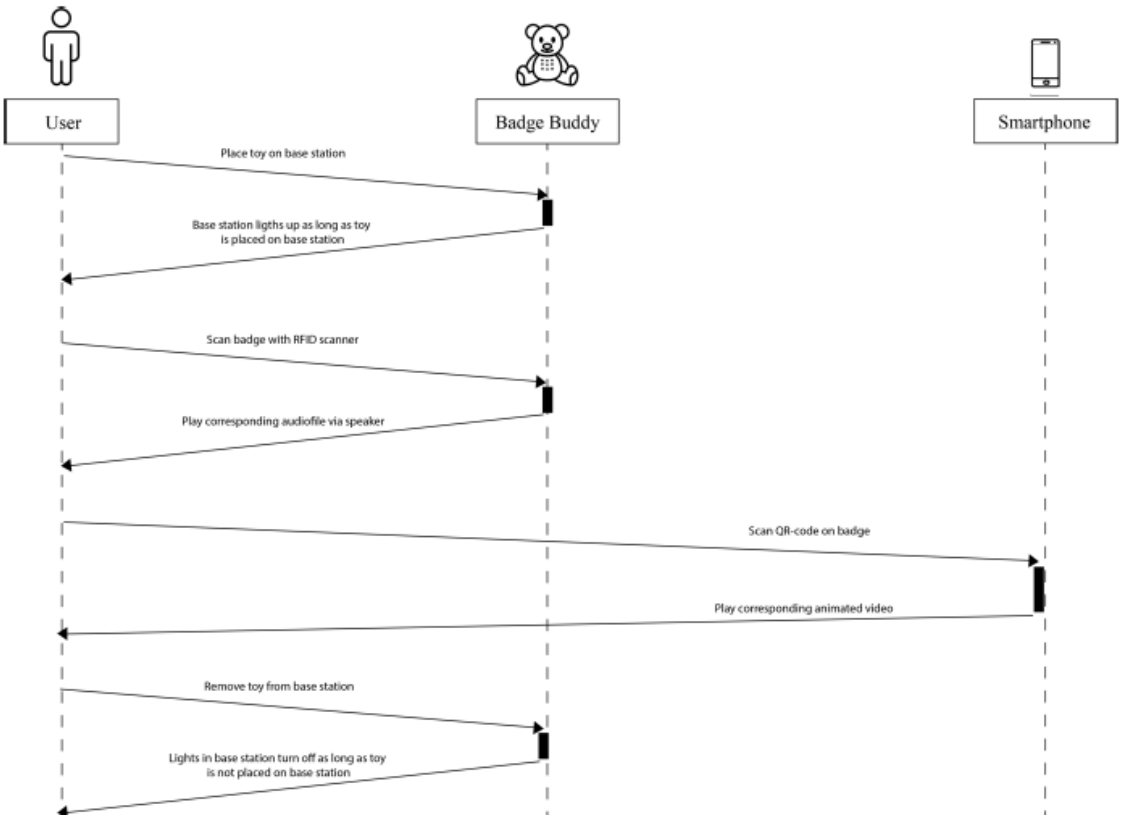


Figure 9.3 A sequence diagram of the Badge Buddy, here time is depicted vertically.

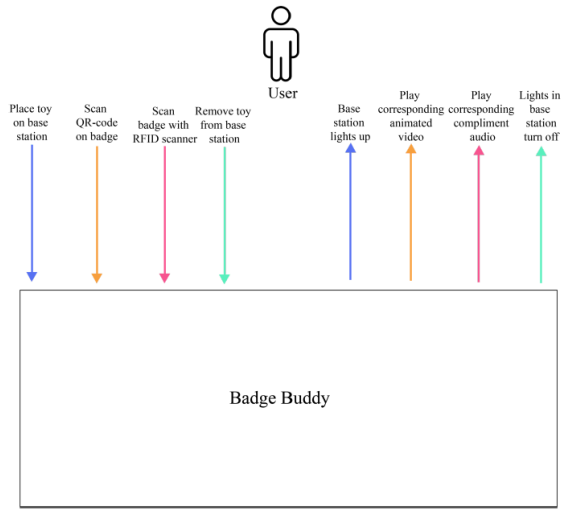


Figure 9.4 Functional architecture scheme of Bade Buddy on Level 0.

Here, we used a time sequence diagram as an intermediate step to assign actions to subsystems. Once we have this we can also more abstractly visualize this in the level 0 functional architecture of Figure 9.4. We can then decompose it to further detail the base station part, see Figure 9.5. Note how the orange line is not part of this decomposition, as this has been assigned to the smartphone subsystem.

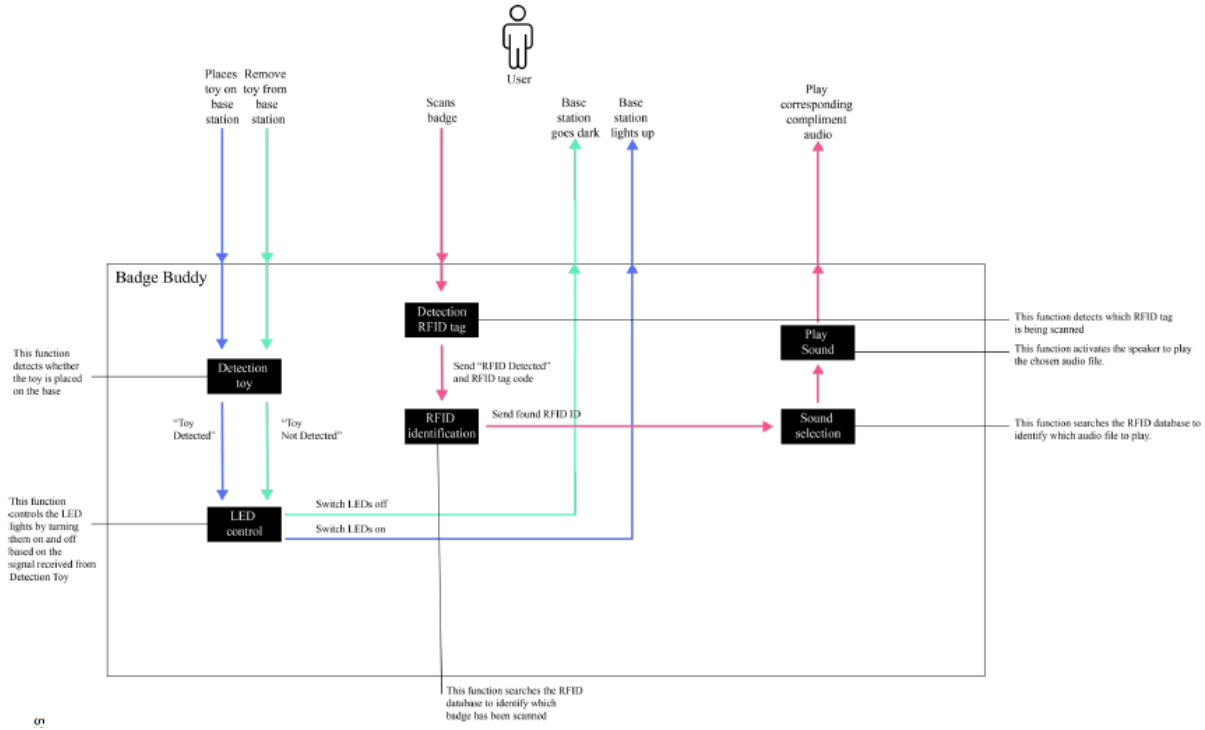


Figure 9.5 Functional architecture scheme of the Base Station sub-system on Level 1

Also, note how the info *RFID tag ID* as part of the function "RFID identification" is used as input into another function to select the sound but does not need to be part of the Level 0 diagram. Depending on who is developing which part with what decision power in the team at what stage in the project, rather than specifying the LED part here already, we might have also described even the sub-component still in rather general means; after all, this might have also been integrated into the sounds or a video. Finally, note that the realization of most "black boxes" are not just software but also have accompanying hardware where parts might be done electronically and parts through software. Similarly, as moving from level 0 to 1, we could have a level 2 of the Detection RFID tag with both an intermediate level of electronic components and pseudo code/functions (rather than in this example, directly going through the realization of an electronics scheme and lines of code, especially the latter document is typically to put in an appendix or online repository, if it is at all needed).

A general rule of thumb would be to decompose a system until it is no longer too complex to implement. Each block should clearly indicate how they relate to each other through their **input and outputs**. Paraphrasing one of our TCS colleagues: *"**models** are nothing more than a simplification of the real world"*. To represent the world, we draw a boundary where relevant input and output are represented on a fitting level through the arrows. For instance, in the Level 1 decomposition, we might have drawn an arrow indicating "power" or a more detailed "5 Volt, .. Ampere power source", and if relevant, besides the light and sound output arrows, also one with "heat" going out. However, here, heat and the type of power were probably not an important requirement, so this was not included.

Functional requirements, non-functional requirements, and constraints

For our running example, in their GP report, similar to your project's report, the first global requirements were related to the stakeholder interviews and the concept ideation, discovering the core

functionalities, and providing this link in a table of the global requirements. Then they showed the graphical representations discussed above with accompanying "development-oriented" rationale and explanation. Only then was there enough understanding to follow this with a set of more detailed functional and non-functional requirements. After which, a chapter on realization followed with the actually chosen components, a description of how each sub-system is implemented with a link to electronic schemes and code in appendixes but clearly showing the photos of the end result, etc. After this, the report is finished with both system tests verifying the requirements and accompanying user tests where needed. You do not need to follow this exactly, and it is very project-dependent, but we do need this layering of levels of growing detail and grounding of decisions.

Literally in the middle of all this are the requirements. In Chapter 4, we already introduced how ***functional requirements*** indicate what actions the product shall satisfy (e.g., *the product shall produce a schedule of all roads upon which ice is predicted to form within the given time parameter* (p11*,* Robertson & Robertson 2012*)*). These were contrasted to *usability requirements*. Usability requirements are only one type of the broader category of ***non-functional requirements*** which indicate in general how well, easy, or fast the system shall be. Besides usability, non-functional requirements categories also include performance, scalability, security, and maintainability. A third type of "requirements" are ***constraints*** that set global requirements of a project, in some cases by company policy or budget set constraints: "*The product shall operate as an iPad, iPhone, and Android app"* or *"The product must be available at the beginning of the new tax year. (p11,* Robertson & Robertson 2012). Keep in mind that such constraints must be built on actual underlying reasons and not just assumptions. Furthermore, it is easy to get a mismatch between the intention of what and the interpretation of how. For instance, the constraint above about iPad, iPhone, and Android app seems to indicate that web-based applications are not good enough. It also does not specify which version, leading to possibly detrimental effects for both the developer (needing to build cross-platform requires a lot of additional work) and customer (they might need to learn to work on two different versions with slight difference). Whereas the customer setting up this requirement might simply have meant they prefer an app that should run on certain devices. Even then this constraint is not yet clear as it would be hard to provide compatibility to all these devices, including very old versions, in turn leading to additional non-functional requirements about maintenance.

Below the functional requirements are included (some rephrased/added in cursive, warning not all are formulated well enough in a ***verifiable*** way):

F1. Should turn on LEDs in base station once toy is placed on base station.
F2. Play sound via speaker when RFID badge is scanned on base station.
F3. Play animated video on smartphone when QR-code of badge is scanned.
F4. Randomize which audio file, out of three options that correspond to a particular badge, is played when an accompanying badge is scanned.
F5. Play only one corresponding audio file within *1 second* when particular badge is scanned
    a. Stop playing audio after this file is done.
    b. Stop any playing other audio file before playing the new corresponding sound. ***[NOTE F5 wrongly overlaps with F2, see next section on formulation!]***

Non-functional requirements (here using NFR for easy cross-reference) included

NFR1. Use accurate medical information on DIPG, its corresponding treatment and procedures in the hospital and at home.
NFR2. Make child patients feel encouraged to go through treatment and emphasize that they are very brave.
NFR3. Appeal to child patients aesthetically.
NFR4. Provide information that can be accessed on a voluntary basis.
NFR5. Should be accessible from a bed as child patients spend much time in bed.

NFR6. Should be easy to interact with for children and adults.
NFR7. Should be easily accessible and transportable at any time.
NFR8. Should include informative and educational animated videos focused on the information needs of child patients by explaining what a treatment/procedure is, how it works and why it is necessary.

Although not used and formulated as such in this particular GP report, constraints included
C1. Electronics and software of base station should run on a Raspberry Pi 3.
C2. A personal stuffed animal of a child shall be used.
C3. The product shall be ready for domain expert testing with four adults in week 9 of the Graduation Project.
C4. The product shall fit within the given budget of the stakeholder plus €30 worth of personal equipment, assets, and similar investments.

C1 was related in the narrative to NFR7 in relation to its form factor, and ease with which functions of the base station (F1,2,4,5) could be satisfied. C2 was explained earlier on but also is a clear system implementation constraint. Typically, constraints like C3 and C4 regarding budget in time and money help to scope the project and solution.

Communicating Requirements: formulation for testing, prioritization, interrelatedness, and contracts

If we would delve deeper into the proposed requirements above, one might critique the formulation of F2 *"Play sound via speaker when RFID badge is scanned on base station."*. We would not be happy when the level with which the sound is played is too low to be audible by the child although in its current formulation it would satisfy this requirement. There might be issues with the .wav or .mp3 recording or with the level of audio coming from the perhaps too cheap speakers. We could improve the formulation regarding the ***verifiable*** aspect (or according the ambiguous but more often used SMART acronym the measurable aspect).

F2.v2 *"Play sound via speaker when RFID badge is scanned on base station within 1 second in an audible format for child standing 1m away."* We could then further specify "audible" as the child being able to repeat 80% of the (key)words of the positive feedback voice recording. This in turn makes clear we also need to specify "the child", here the child is a Dutch-speaking child within the age range from five to ten years old, so we would further specify F2.v3 *"Play sound via speaker when RFID badge is scanned on base station within 1 second where 80% of the content can be repeated by 90% of the five to ten year-old Dutch-speaking child-clients standing 1m away.".* The percentages have to be aligned with a source, for instance in this case the stakeholder might agree on this during an interview, or user group research maybe show that a certain percentage of the target group would be too hard to include reasonably. The wording client might be preferred over patient in some cases. We then have to also align our user tests where this one clearly indicates a test with our user group, whereas some others might be simply a system test where the testing team themselves for instance does 10 repetitions of placing and removing the toy.

As indicated in Chapter 4, besides looking at the characteristic of being verifiable, the checklist of requirements by Ivy Hooks also indicated **necessary, verifiable, and attainable**. The **necessary** criteria made us remove a requirement about turning off LEDs by simply rephrasing F1. Depending on the project, we might have also rephrased the audible part with an additional NFR in relation to F2 rather than formulating F2.v3. In general, it can be hard to distinguish NFR from F. Should we see randomization as a subset of F2 indicating how well the sound played is not annoying and surprises the child? Or rather is it the verb randomizing with which we put the function more centrally in the design?

For small projects, it might make most sense to include properly detailed requirements (including NFR parts making it testable/verifiable) rather than adding too much relationships between functional and accompanying non-functional requirements, or to spend too much effort on splitting them perfectly for such edge cases.

From the SMART criteria[11] the original A implied ***Assignable***, knowing who or what does what, which we have not yet covered. We did already discuss how sub-systems need to be related to each other to know which data needs to be transferred from one part to another (cf. public classes and input and return format of functions in coding). When looking closely to the formulation of the requirements we see the requirements are actually assigned to the visualized subsystems: F1, F2,F5 to the base station sub-system (and reformulated in relation to the user and F5 related to the badge), F3 to the smartphone sub-system, F4 to both the stuffed toy in relation to the base station.

Prioritization

To keep an overview of all design decisions and envisioned 'ideal' concept while keeping the system **attainable,** the **prioritization** of requirements can be key. Robertson & Robertson (2012) also relate priority to release moments of a product: *"The priority of a requirement indicates the importance of the requirement's implementation relative to the whole project and governs which requirements will receive priority for development for the next release of the product." P364*
One well known system for prioritization is **MoSCoW** : Must have, Should have, Could have, Will not/won't have. The must haves have to be satisfied and are its core functionalities, the should haves make the system function better and are probably needed to really role it out, the could haves are more the gimmicks and the nice to haves, and the will not haves help to scope the project to refrain regarding its constraint from implementing certain unnecessary tempting elements. In our example, F2 + F5 (play rewarding audio) and F3 (video) were identified as must haves, F1 as a could have (whereas the turning off part was indicated as a should have, perhaps identifying that wrong information about the system's state is worse than giving no information).

Another system can be simply ordering them from highest to lowest. Yet another would be to assign 1-5 or a 1-99 rating, where it is important to mention whether 1 indicates highest priority and 5 the lowest. This separate prioritization from ordering might help if requirements are ordered to group them in sub-systems or to make a more understandable narrative. Sometimes requirements can be grouped in relation *"to use case, component, features, or ... other ... unit",* perhaps even personas, which requires grouping them together (p382), as often some functions only work or can be released if a set of requirements are fulfilled (if security or privacy is not in order we might test with preset scenarios or safe environments but not release such a version).

Factors for prioritization also include (Robertson & Robertson (2012), p382): cost, value to client/customer, ease of/time for implementation, ease of business or organization implementation, benefit to business, or obligation to obey the law.

To substantiate your prioritization, one might use stakeholder or end-user interviews, focus groups, or card-sorting (Robertson & Robertson (2012) also discusses voting or €100 monopoly money budget division) and build a ***prioritization spreadsheet*** where ratings/scores on ease and cost of implementation and value are combined with weights to come to a 1-10 priority rating per requirement, indicating where to work on most and/or first.

[11] Specific Measurable Assignable Realistic Time bound; often the A is replaced by achievable/ attainable. There are many variations for these letters, like reasonable or results-based for the R, if look at only those mentioned in the Wiki we can come up with 2160 variation, so ironically SMART itself is not smart, as it is not specific 😊.

Conflicting and combinational requirements

After **prioritization,** we might also need to resolve any **conflicting requirements**. Those are requirements that can't be satisfied simultaneously. For instance, one constraint requirement might indicate the system needs to be finished within a month, but another requirement asks for a component that has a longer delivery time.

Robertson & Robertson provide a few pointers to spot conflicting requirements through sanity checks:

- Requirements that use the same data (search by matching terms used)
- Requirements of the same type (functional, non-functional, and constraints search by matching requirement types)
- Requirements that use the same scales of measurement (search by matching requirements whose fit criteria use the same scales of measurement)

Besides conflicting requirements, we might also have ***combinational requirements*** where one requirement has to be satisfied with another, and thus tested together with another as very well illustrated by the example of the Coast Guard case where speedboats. In these case the boats were not build to go fast and robust simultaneously but only satisfying these requirements independently, see text block below:

Rijkswaterstaat wasted one milion euro on useless speedboats
According to "de Volkskrant", Rijkswaterstaat made a mistake in the tender. The service incorrectly formulated the performance requirements that the speedboats had to meet. A so-called combination requirement was missing from the documents. It said that ***the boats had to be able to sail at high speed*** *and* ***through high waves****, but not that they had to be able to do this in combination. Van Nieuwenhuizen writes in the letter to the Chamber that the amount cannot therefore be recovered from the supplier. The vessels are too large to be used on rivers or inland waterways. Rijkswaterstaat tried to auction the boats last month for a "bottom price", but that failed.*

Google translate – " " added, NOS Nieuws Tuesday 21st of augustus 2018, 03:41

To prevent such conflicts, especially large sets of requirements need to be structured, consider the following for instance, with separate columns:

- numbering,
- grounding/rationale, justification linking all requirements preferably to user needs and research
- relation to other requirements, either to provide the rationale or to indicate combination or grouping in units fitting prioritization, provide internal and external links to go back and forth in the document
- type of requirements (e.g. functional, non-functional, and constraint)
- prioritization, either through grouping, ordering, or a number
- version numbering

Where it is of utmost importance that the wording is verifiable, necessary, attainable, and assignable (e.g., component or development team). Furthermore, in Chapter 4, we already explained how the requirements need to be embedded in the bigger narrative and here we gave a few first communication tools regarding functional architecture decomposition and a customer journey.

The end user as starting point and end point

An important rule of thumb is to have each functional and non-functional requirement linked to an underlying user need. So if we build an app, there might be a non-functional requirement about size in GBs, and an accompanying rationale might include the user need to be on location in the city and download the app within 3 minutes while not going beyond their data budget. Note how this would indicate a need to change related requirements when people would have 6G rather than 4G speeds. This seems to be one reason to use a **Requirements Traceability Matrix**, where besides the explained part above regarding description and justification, there is also a link to any (intermediate) test results.

Without such links to testing and grounding (the rationale or justification of the requirements), it might sometimes also feel counterintuitive or far-fetched for the people who implement them. One can imagine that a well-paid web designer might be surprised if they are asked to provide a website that looks "cheap" or even "unfinished". Knowing that it is about a low-priced supermarket such as Aldi, Lidl, Dirk, or Walmart might already make more sense. However, remember that geographical and cultural differences might make the developers or graphical designers unaware of such seemingly obvious contextual parts. To even put this further in perspective for the graphical designer, it is good to communicate the underlying user need where the envisioned *user experience* is testable, one where the price is not only low but the complete experience also feels like money is not wasted, see Figure 9.6.

Figure 9.6 Screenshot of Aldi.nl, a low-price supermarket, with a surprising lay out including blocking text. Through guest lectures we have heard rumors that such decisions might very well be intentional. Such a design/house style might be tested occasionally on perceived money spent on web design while also rating professionality, attraction to buy, or similar measures. Although unlikely to happen in detail to the graphical design it might help to formulate: the graphics of weekly discount should score a 4 out of 5 on attraction to buy (5 being attracted to buy tomorrow), cost for design should be perceived as below € 500, and score 4 or higher on fit our fit to brand scale (5 being perfect fit for, currently comparing against Lidl, Dirk, AH, and Jumbo).

For websites, we can test or discuss such graphics through various means. One way often used early in the design process is to design and show **mood boards**, that is a set of pictures representing a certain mood. Ideally, in our view, the images are not from the product category itself as we should discuss aesthetics/mood -not the actual implementation yet. For instance, using a set of pictures from clothing styles, houses, and electronic devices even when designing a car website. Different, but clearly providing the idea, most perfume commercials focus on this certain mood before showing the bottle etc.

The use of **color schemes** helps to define graphics, sometimes building on research on the underlying culturally dependent meaning of colors. Another means is style tiles where a small set, like 3-5 printed tiles have a combination of font/typeface, colors, and certain graphic elements (e.g., modern techy or clean toned-down) and can **discuss what the stakeholder or end use**r appreciates and dislikes. These decisions can then be worked out iteratively into (house) style guides or brand identity documents. These are used as specifications for graphic designers and sometimes employees working on webpages,

documentation, presentations etc. For instance, have a look at our own university **house style overview**. In such documents, decisions are communicated practically: "*Contrast with background: In order to maintain sufficient contrast with the background, the black logo is only placed on backgrounds with a grey value of 0 to 40%. For backgrounds with a grey value of 40% or more, the white logo is used."* These documents do not just give the specification but **also why in relation to an end user** (i.e., to maintain sufficient contrast).

For all kinds of products, we can link requirements from text descriptions of an underlying user need to implementable specifications with accompanying rationale. This is not only related to just software or smart products but also holds true for physical and electronic devices. For instance, at a large electronics manufacturer in the Netherlands, even the weight and sounds made by their equipment can be linked through user studies to the accompanying price segment. This also shows that the user is not only the starting point of the design but also the endpoint after the implementation with which the requirements have to be tested (e.g., fitting a requirement that 95% of the users shall perceive the estimated cost of the device between 70-90€). Furthermore, this example shows that we might also **test separate parts** of the device, such as the sound, weight, tactile feel, or looks, without having a complete product. This, in turn, specifies our starting requirements into specifications (e.g., specifying a certain frequency domain for sound) through user testing. We can easily have hundreds of participants listen to potential sounds generated by speakers placed within an electronic toothbrush to further define how a frequency domain might relate to perceived costs, although we have to be aware of potential interaction effects such as the one between color and sound frequency (Wang et al. 2019).

V-model as an organizing methodology: increasing detail and testing.

Ideally, a design process would have an agile approach, an approach where there is a constant reflection and iterative adaptation of requirements whenever new insights arise. In Chapter 4, we already reiterated that, according to the agile manifesto, we would focus on working software over comprehensive documentation and work on customer collaboration over contract negotiation. This seems to only work for small, fast projects or in-house collaborations where documentation can be kept to a minimum. However, often, companies must outsource parts to other (sub)contractors through a "tender" (i.e., a call for bids) formulated in a structured procedure. Especially in governmental institutions, such a tender process must be transparent; what is required by the institutions? How do the companies intend to satisfy these needs, in other words, what is their bid? During the execution, there might be only a limited budget for changes or joint discussions informing the design decisions. Furthermore, whether the end result of the company then fulfills these requirements might also become a question of debate.

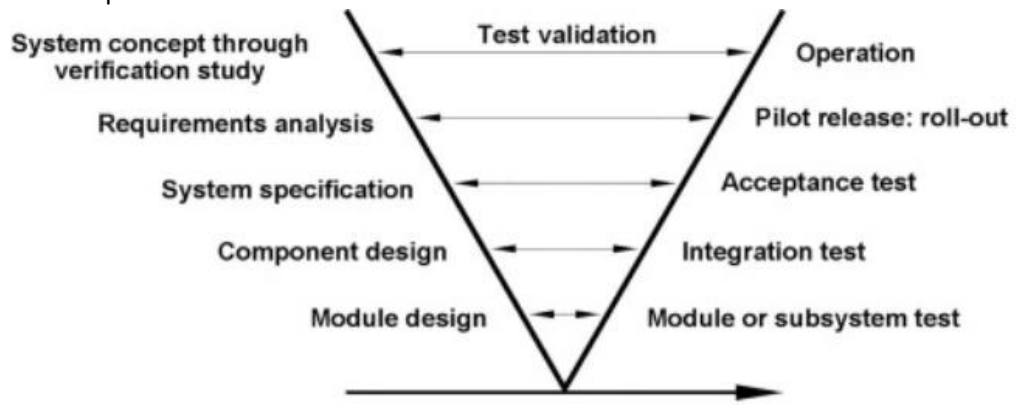


Figure 9.6 Description of the V-model https://www.sciencedirect.com/topics/engineering/v-model.

Churchill once reiterated that *"democracy is the worst form of government…. except for all those other forms that have been tried."* Similarly, although far from ideal, we can learn from the "V-Modell" used by the German government, see Figure 9.6.

In the V-model, we build from general concept descriptions to requirements and specifications, which are then worked out in code in the end. On the right side, we see a form of testing on each level. Note how we previously indicated how one can test separate elements such as sound and weight. In practice, the company performing the tests could be required to be completely independent from the company designing or delivering the requirements or specifications to prevent influences and keep a transparent process. This is common practice in some countries, for example for the government projects in Germany. Furthermore, multiple companies—or at least different groups of people—will be involved in the vertical levels where detail is added. For example, it goes from a general description of a toothbrush to a printed circuit board made out of many even more detailed components and connected to other subsystems, such as a functioning motor. When setting up requirements, it is a good idea to consider the requirements you make both from your designer perspective as well as from a development/engineering **alter ego:** would you come up with other solutions that also satisfy the requirement? Would you be able to understand the requirement, given the surrounding narrative?

This means that the work needs to be specified and, in the end, engineered from one layer of detail into the next more detailed layer, decomposing or "unboxing" parts of the system in each step while keeping a clear view on how elements are related. This needs to be communicated so that both parties understand what is needed and why. In recurring partnerships, it is not unlikely that requirements that are set up for a tender are questioned by the developing party to discuss whether they can be less restrictive in order to overdeliver in another part. However, if the V-Model is followed in a time-sequenced manner involving many parties, this might become impractical leading to even more importance on the detail of documentation.

In the previous examples from graphical design and electronic devices, we already saw that, also in smaller projects, we can use different layers of detail with varying tools of communication (house style vs frequency spectrum) dependent on the domain of the product.

Ideal world, manufacturing, and prototyping

Often, CreaTe students mistake their prototype as suitable for testing with an envisioned end product. In some cases, for instance in an immersive VR experience with existing hardware, these two might be quite close. However, in others, these will be very different; it is unlikely to see laptop computers hooked up to remote controls, nor would they have an Arduino or Raspberry Pi inside. Instead, the requirements in most student reports are those that have to be satisfied to test the prototype to make the claims needed for whatever step of development the project is in. In module 6, this often includes a technological feasibility claim aligned with a technological opportunity by combining off-the-shelf hardware and some electronics, as well as working out the aesthetics and interaction into a resulting satisfying user experience underlying a certain identified user need. The devices, materials, and manufacturing procedures we chose for our prototypes are different from those for mass production and single devices released in real-world critical situations. Some parts might not be rugged enough to be used for a reasonable life span, others might be too expensive to reasonably produce. Scaling up production often happens in steps; on this journey, it can be convincing investors or other financing bodies, and once intermediate results are promising, the number of units produced per batch goes up and will require or allow for different manufacturing choices.

In one Interaction Technology master thesis project, a student worked on a STEM-promoting workshop with an electronics toolkit that could measure when connections were incorrect. The toolkit guided the users through making the correct electronics circuit, while connecting these components to an Arduino step by step, resulting in an educational box and accompanying workshop. In this project, we can nicely see how one can work from global requirements, here called intermediate statements, to identify they do not satisfy the criteria of properly formulated requirements to the ideal specifications and simplify

them down to the final specifications to be used within the projects' constraints (here time but sometimes also budget to develop compared to cost per unit).

Intermediate Statements	Initial System Specifications	Unit
LSB will measure time	Real time clock with a resolution of 0.1	s
	Accuracy of 100	parts per million
LSB will specify tests to perform on a DUT	LSB provides test vectors to the DUT	list
LSB will allow the user to select tests for the DUT	LSB has a capacitive touch screen of letters	list
	LSB has a menu select device	quadrature encoding
	LSB has a menu select button	binary
LSB will record and report number and type of faults	LSB will record test vector results from the DUT	list

Final Specifications	Unit
Real time clock with a resolution of 0.1	s
Accuracy of 100	parts per million
LSB provides fixed test parameters to the DUT	list
LSB has an LCD display 2rows x 20columns	list
LSB has a menu select device	quadrature encoding
LSB has a menu select button	binary
LSB will record fixed test parameter results from the DUT	list

Figure 9.7 Snippet of text of a master thesis project on the Learning Box System showcasing how one iterates through different requirements and specifications depending on the phase. In an additional final overview, the component brand and type were specified that satisfied this list.

Conclusion

How to formulate requirements properly implies they are necessary, verifiable, attainable, and assignable. This form should clearly show how each set of requirements is measured. To communicate them throughout a design team, they need to be structured, forming a narrative, aligned with the previous steps of gathering them and the future steps of testing, for which a matrix with rationale (users & research), tests, relations, and prioritization is helpful. Through prioritization and coupling them with other system visualizations, they clearly identify the subsequent design steps; together, this indicates a bit of the comprehensive, properly legalistic hurdle requirements engineering can be.

Further Reading and References

We only touched upon some types of modeling user behavior, system actions, and data structures. See the wiki on Unified Modelling Languages https://en.wikipedia.org/wiki/Unified_Modeling_Language. specifically the table overview of UML diagram types. We, for instance, discussed the (time) Sequence diagram, but you might also be familiar with a State diagram and the flowchart like an Activity diagram; another interesting diagram might be the use case diagram when multiple types of users interact with a system, such as an ordering app. Other resources used or interesting to read are:

- Dix, A. (2003). *Human-computer interaction*. Pearson Education.
- Robertson, S., & Robertson, J. (2012). *Mastering the requirements process: Getting requirements right*. Addison-wesley.
- Wang, L., Qian, D., & Li, O. (2020). The cross-modal interaction between sound frequency and color saturation on consumer's product size perception, preference, and purchase. *Psychology & Marketing*, *37*(7), 876-899. [random example of interaction effect on customer preference, e.g. *"only when the color of the product is in high saturation can the sound frequency significantly influence the perceived size"*]
- Kundan Nandlal, The Power of Customer Journey Mapping in Product Development, LinkedIn post, 21-8-2023, last accessed 3-1-2024
- Hillebrand, M. (2021). *How to effectively communicate brain stem cancer to a child patient* (Bachelor's thesis, University of Twente).

- Nonfunctional requirements, Scaled Agile, Inc., 13 October 2023, last accessed 3-1-2024. Note: an overview and explanation of Non Functional Requirements,
- Alistair Sutcliffe, 2014, *Requirements Engineering*. Interaction Design Foundation - IxDF. https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/requirements-engineering

Part IV – Making beyond Research

8. Implementation

Note that the chapter is no longer part of the materials for theory but is useful for the project.
As part of the user-centered design of hi-fi interactive prototypes, we need to implement intelligent interaction technology. Previous modules taught parts of this. Using the skills learned in M5 and M1-4, it is possible to make models and implement systems beyond simple single trigger-response patterns.

Furthermore, apart from the interaction models, it is necessary to develop the interface and controls through which this interaction is driven. Such things can be made stronger and more effective by building upon design principles, heuristics, and patterns. This chapter of the reader provides a few entry points to that.

Third, besides the underlying models and algorithms that control the interaction, and the interface through which they are driven, there is the actual *content* that is delivered by the system. The design of an interactive learning system includes learning content; the design of interactive sports exercises includes training content; the design of information-providing agent includes conversational content and access to the actual information that is provided; and so on.

Finally, the design and implementation of the hi-fi prototype need to be documented. Besides explaining the above aspects of an intelligent interaction technology system, this involves explaining an overall architecture – that is, the decomposition of the system into components that work together, paying attention to things such as where the input, the output, and the decision-making happen.

Some thoughts on more detailed design of hi-fi prototypes

There are various resources on the web and other sources of information that can help you in the design. Some are very concrete, others much more abstract. Which resources are relevant for you depends a lot on the actual system concept of your project. In this section we will provide some directions, while being aware that coverage of the field is not complete.

Design principles and heuristics

Design principles or *heuristics* are abstract principles that, when followed, can lead to better interaction design. They are abstract, in the sense that they do not help you make specific decisions on details of your system, but nevertheless they can be helpful. Design principles can be studied before working on the design, but also can be used to validate some of your design decisions in retrospect ("how well does our system comply with this principle?"). Some examples:

A principle of *feedback* may state that when a user takes an action, the system should give feedback to indicate whether the action was successful and what the result was.

A principle of *affordance* may state that interface elements of a system should provide information about how they can be used (that is, for example, why GUI buttons often look like real-world buttons).

A principle of *familiarity* may state that one should match the design with the experience and expectations of the user.

A principle of *accessibility* may refer to something difficult like giving blind people access to certain tools, or something as simple as taking into account color blindness (about 4.4% of Dutch population[12]).

[12] https://www.accessibility.nl/kennisbank/artikelen/kleuren/kleurenblind

A principle of *sensorimotor factors* may dictate that you design certain aspects of your system in a certain way *because human perception and motor control inherently are stuck with certain rules*. For example, **Fitts's law** says ***"The time to acquire a target is a function of the distance to and size of the target"***. This is a sensorimotor law: it is determined by how human motor actions and perception are coupled and constrained. A good explanation of this, and its ramifications, can be found at the first link under "further reading" below and we will discuss this in the lecture.

You can use rules and laws and principles such as these to decide where to place icons and menu items, how to order your GUI elements, whether to sort things top to bottom or the other way around, but also to decide how to organize embodied and tangible interaction.

Design patterns

Design patterns are much more concrete than principles and heuristics. A well-known concept from computer science, design patterns describe a general solution to an oft-occurring problem. Design patterns develop in practice, through repeatedly solving similar problems and finding similar solutions. Most computer scientists know the software design patterns (e.g., using interfaces, delegates, factories, facades, server-client structures, etcetera). For this project, we are interested in design patterns for *interactive interface design*. There are quite some examples of design patterns for menus, input forms, navigation principles, …. In the further reading below, some sources are given. When you are working on specific GUI design tasks, it may be useful to have a look at possible relevant patterns for your task. If you use such patterns, don't forget to refer to them in your report.

There are also numerous sources on principles, patterns, and heuristics for designing visual communication, visual storytelling, and use of sound and imagery in interactive systems. It is often worthwhile to search for such things for whichever project you are doing at the moment.

Further reading

Here is a list of some further reading. Some of it may apply to your project; other things might not. In any case, none of the sources in this list are part of the exam; they are just presented here in case they help you in the design of your system.

- Very, very nice website, regarding to Fitt's Law, easy to read and fun to do part of reading materials: ~~http://asktog.com/atc/principles-of-interaction-design/~~, often broken, if so try [link], also see Tog's IxD first principles [link]
- A paper by Eva Hornecker and Jacob Buur on design guidelines for tangible interaction and shared workspaces. It is harder to read and very interesting, but leave it alone if a quick scan does not convince you of its relevance for your project or your personal interest. Hornecker (2005) A Design Theme for Tangible Interaction: Embodied Facilitation. In: Proceedings of ECSCW'05. Available as download *[link].*
- A paper by Wendy Ju on the design of Implicit Interactions (systems that serve proactively, in alignment with our human tendencies mapped to products, yet keeping these products on the background). Interesting, easier to read than Hornecker's paper, but do a quick scan first to see whether it catches your attention. Ju & Leifer (2008) The Design of Implicit Interactions: Making Interactive Systems Less Obnoxious. Design Issues: Special Issue on Design Research in Interaction Design, 24(3) Available as download from *[link]* (search for the title).
- A paper by Floyd Mueller that gives a design taxonomy of "exertion games" (a term invented by him!). Short, easy to read, especially section 4 & 5 are useful for some projects. Mueller & Isbister (2014) Movement based game guidelines, In proceedings of CHI'14. Available as download [link].
- DINED is a practical tool for knowing and dealing with measures of the (mostly Dutch) human body https://dined.io.tudelft.nl/en (e.g., grip circumference, height, and reach)

- www.welie.com/patterns (mostly GUI patterns)
- www.ui-patterns.com (also contains patterns for persuasive design, social systems, and game mechanics!)
- http://www.gameplaydesignpatterns.org (several hundred game design patterns)
- Possibly, other papers may be added on Canvas or in the course manual that present design guidelines for a system of the type that you are currently developing; feel free to ask.

Part V – Measuring Success

9. Product validation by results

As we have read so far, a typical HCI project in an education and research setting often goes through similar stages with recurring activities (observations, illustration, confrontations, and implementation). The projects that come out of this can be extremely diverse.

In the first phase, we delve into the literature to find a grounded rationale. This can be done in two ways; the first is building on the theory of why a certain approach might work. The second option is that we look for state-of-the-art and related work in engineering; in this approach, we might build on a body of similar systems, studies, technologies/patents, or observations; and, from that context, build our expectations and distillations of what might work. In this way, we use previous results to explain why the system works in a certain way.

At the same time, this also means we need a clear vision on how and why we go beyond what is currently out there. Besides knowing why something might work, we also need to be sure that our product actually will add to the lives of people, and often, this is done via iterative user confrontations and final evaluation representative for actual use. Many successful projects combine existing technologies in new and creative ways to address new problems or new contexts. Clever choices in what to reuse from others while crediting them can be a decisive factor in whether the project fails or succeeds.

As soon as we test a system, there is a healthy tendency to compare the system to an alternative, where we systematically introduce the developed system or alternative with random assignment. The book by Lazar et al. provides more detail about the random assignment, either choosing to use different participants (between-subject) for each condition or when using the same participants (within-subject), switching the order of conditions. For the latter, the Latin Square Design provides a nice fair way to switch orders between participants. However, the second choice is just as relevant. How to decide **what conditions to choose?** Often, the question at the end is, what alternative should be compared to?

The first and easiest and not-so-satisfying option is to test the design against **nothing at all**. This is when people build something and just mention what effects it has. For instance, when building an immersive VR environment for learning Japanese words to simply state how much words were correctly identified after playing with the system. Still, within the same context, a bit more complicated measure could be to also look into retention, whether after four weeks or so the words are still remembered.

The second easy option is when there is a fair comparison of the designed product to **an existing baseline**. For instance, when designing such an educational tool, comparing it to the current way (e.g., a lesson including teacher and book) could form a good baseline to compare to.

The third, and sometimes most interesting approach, is to show why a system outperforms other alternatives by **adding or removing the specific new interaction element**. In the case of the running educational VR system, the student knew already that VR could help in the acquisition of words from the related work and how it compared to various other methods of learning. So rather than again proofing this effect, but with their specific system for their specific language with their specific participants, they looked at one particular new element. In this case, the use of varying the representation and location of the items linked to the words, versus varying only the location of the items. Note that in an educational setting, the time spent on a task is often something to take into account and keep fair.

Investigating measures - efficient, effective, satisfaction, and UX.

In some cases, the measures that can be used to compare systems are rather straightforward and context-dependent. For instance, in an educational setting the number of words remembered can be a

meaningful outcome measure. This can show whether a product is ***effective***, can we do with it where we designed it for. Another well-known measure is whether a system is ***efficient***, how quickly can we do the things with the product where it is made for. The third standard outcome measure is whether users are ***satisfied*** with the outcome. Some tasks that we can do within a short time do not leave a satisfying feeling.

In other cases, and basically mandatory in this project, we need to investigate a specific User Experience that we aim for. These are often harder to measure.

For measuring UX and other related constructs, creating a good questionnaire takes too much time for almost any project we embark on. Luckily, to this end, there is a variety of already existing questionnaires that are used in academia and industry. For instance, regarding usability, there is the `quick and dirty' System Usability Scale (SUS by Brooke 1986); notice that their scale goes from 0-100 but does not represent a percentage, nor is 55 a passing score; rather, 68 is seen as the average. Although we do not require to know each of these, a few other "questionnaires" which we often see and use in our education and research and can be of use for several GPs and M6 projects are:

- QUIS, Questionnaire for User Interface Satisfaction (QUIS) by Chin et al. (1988) contains five traditional usability scales: Overall Reaction, Screen, Terminology/System Information, Learning, and System Capabilities.
- The USE, usefulness, Satisfaction, and Ease of Use (USE)
- The PXI, the player experience inventory intended primarily for game design. (van den Abeele 2016, ….) We suggest to use this one and not the various questionnaires called Game Experience Questionnaire (most well-known Ijsselsteijn et al 2008.).
- Immersive Experience Questionnaire (IEQ) by Jennett et al. (2008) which is used for testing feelings of immersion.
- Intrinsic Motivation Inventory (IMI), contains various sub-scales most relating to the Self-determination theory (relatedness, competence, perceived choice, interest/enjoyment, effort/importance, pressure/tension, and value/usefulness). Most papers are also by Ryan and Deci, it has clear instructions for use but needs a license for commercial purposes.
- The Godspeed questionnaires by Bartneck et al. (2009) are used to investigate mainly robots on how human-like, animal-like, likeable, intelligent, and safe they are.
- The Borg scale, measuring perceived exertion (Borg and Linderholm 1967)
- The NASA TLXScale, used to measure task load (cognitive load or how demanding the task is).
- The IOS, unlike most other tools on the list the Inclusion of Other in the Self is a one item measure by Aron et al. 1991/2. It uses overlapping circles to let people indicate how close they feel to another person (or with some adaptations a system).
- The Self-Assessment Manikin, created by Bradley and Lang (1994) similar to the last tool uses five pictures to measures pleasure, arousal (i.e. the feeling/level of excitement), and dominance.
- The fun toolkit by Read and MacFarlane (2006) is a tool for measuring "fun" with children, it includes several tools including the again-again table, the fun sorter, and the smileyometer. The accompanying paper describes the use and considerations (also mentioned by Lazar et al.)
- *https://hci-studies.org/ from the materials on Canvas, here you have to click on „Methods and Measures" from the Toolkits section. The paper can be found here https://dl.acm.org/doi/pdf/10.1145/3544549.3585890*
- https://www.asarif.com/notes/scales.html but this also includes some not validated questionnaires (such as the GEQ by Ijsselsteijn) so perhaps more an overview of what is used, than what should be used.

The pragmatic balance between validated and attainable questionnaires

Sometimes, the validated questionnaire has items that would not work for your study; imagine

investigating a new interactive educational tool that replaces something that previously was always done on paper. The best questionnaires then often still have a descriptions such as “This book felt like a fellow student wrote it.” This formulation will not work for your system, and then adapting is needed. However, do this adaptation precisely and only for that part of the questions.

Another important issue is that asking questions influences the thoughts of the participant. So if we first ask questions about milk, cows, refrigerators, small animals, babies, and white walls, and then we ask, “What *does a cow drink?”* we might think it is milk rather than water. Similarly, reordering questions of a questionnaire and taking out or adding questions (very likely) does not only influence one question or even one construct but probably multiple. So, always check how the questionnaire is validated. One question to check is whether they randomized the order, as that would indicate that the questionnaire has stability in the results against such reordering influences.

Another element the authors might discuss that relates to the influence of one question on another is whether they prescribe that you use the separate constructs of the questionnaire. If separate constructs are allowed, this can help a lot: 1) in how long the test takes for the participants as there are fewer questions, and 2) it helps us remember to focus clearly on what exact part of the experience we target. The latter also influences how likely we can get a significant effect from the chosen statistical analysis methods. To compensate for doing multiple tests (for instance, when you do have to include quite some constructs), often something like a Bonferroni correction is used to prevent the results from being found by pure chance, which means that you have to divide the found p-level by the number of tests!

Similarly, if you need a translation of the original questionnaire, there are a few steps you should take. The easiest would be if someone else translated it for you and validated it. If not, you no longer have an officially validated questionnaire, but we can add rigor by using the following techniques. Use back translation, so you translate the questions and then let someone else translate them back to their original form. You can verify whether nuances are kept or if other terminology might be needed. If you are graduating and there is no one else to help you out, sorry to hear this, the next best thing might be using translation software to do the same back translation.

So, you should edit nothing of the questionnaire unless you really can not use the current formulation. Check the related paper to follow the instructions about the order of questions, the constructs to be included, etc.

Analysis

For most traditional scales (including the first five of the overview 2 pages ago), the questions have so-called constructs within the questionnaire (e.g., learning and relatedness). Having various questions to measure one such construct seems to help in getting people to understand the measure in the same way; rather than just directly asking about your level of relatedness, it can help to ask specific questions together describing a rather broad, specific, or peculiar topic. For each construct, a set of about three to seven questions must be answered on a scale that can then be averaged. The scale used per question is either a Likert scale where an agreement to a statement is measured (e.g., 1 = Strongly Disagree to 5= Strongly Agree, preferably on a 3/5/7 numbered scale providing some granularity) or a semantical differential scale (words spanning two ends of a spectrum such as failure / perfect on the TLX), in both these cases we call these descriptive words on the scale “anchors” (e.g. the Strongly Disagree).

The researchers behind good questionnaires already checked that they measure what they want to measure (meaning it is ***valid***) and that these measures function consistently, getting the same accurate response (***reliable***). Nonetheless, it is good practice to report the Cronbach’s alpha of the original study; this indicates whether the set of questions under a construct has consistent responses. If a person answers one question/statement under the construct with a high value then we expect them to also

score high on the other questions. For example, Vanden Abeele (2020), in their PXI, uses the construct *Challenge* with the statements "The game was challenging but not too challenging." and "The game was not too easy and not too hard to play." (Table 8). Often, we also see the Cronbach's alpha of the collected data of the study although there are other more relevant measures it seems some researchers expect it (see Lazar chapter 1 about difference between disciplines).

We often need to know what forms of statistics to use, and for this, we need to know what kind of data there is. There are three main types of scales: nominal, ordinal, and continuous.

- Nominal data, sometimes called categorical data, includes the country of origin (e.g., Netherlands, Germany, South Korea, UK, US…); there is no clear order in this type of data.
- Ordinal scales include a form of order, but differences are not equal, for instance, when asking about age with predefined categories <18, 18-64, or 65+.
- Continuous scales have a meaningful equal distance between values (e.g. distance in meters). Continuous can be split into interval (temperature in Fahrenheit) and ratio (true zero such as mass with 0 kg or temperature in Kelvin). The latter difference impacts certain claims and what can be done with the data; for instance, one cannot state your product is x times better on an interval scale.

When working with one single Likert-scale-formatted item, it would be ordinal. However, when averaging multiple values, we start to assume that differences in different statements/questions are equal, and with the various values it can take, it starts to resemble a continuous value. Practically, it is, therefore, often analyzed as if it is continuous data. This means that if there are enough participants we see that often a t-test is applied when using Likert-scale data as an outcome measure, although mathematically, we break a few rules (probably not constant distance between values 3 to 4 compared to 4 to 5 on a scale from Strongly Disagree to Strongly Agree, and values are bound on the left and on the right). Even in these cases, when data is continuous, and we assume it is normally distributed, researchers do often include a check for normality to test whether it is not significantly different from a normal distribution.

		Independent Variable		
		Nominal	**Ordinal**	**Interval/ratio**
Dependent Variable	**Nominal**	Chi-squared test	Choice model	Choice model
	Ordinal	Mann-Whitney	Spearman-rank correlation	Ordered choice model
	Interval/ratio	t-test, ANOVA Linear regression	Linear regression	Linear regression

An overview to the recurring types of statistical analysis, credits to Dasha Kolesnyk

For exact ways and explanations on how to do statistics and how to report statistical analysis, always check the specific test or reporting standards, as statistics is not our field of expertise. However, a few practical pointers can be useful as we often see suboptimal reporting. Besides the elements before it is important to realize it is not enough to just report a p-value (the chance of it happening by chance). Something can be very significant but might not be very meaningful if it has a very small effect. For instance, when selecting a certain icon in a task is only used a few times a year, it is not very meaningful

if there is a very systematic increase of just 0.01 seconds, even if the p-value is very low. Therefore, reporting how much the values are influenced is good practice. Depending on the data type, this can be a median (middle value), perhaps accompanied by quartiles or box plots, or an average, which is often accompanied by a standard deviation for a normal distribution.

The end,
we hope you had a good experience with this reader!