



FH Salzburg

Playful Onboarding in Software Development Projects

Exploring Play in Digital Onboarding Processes

Master's thesis for the award of the academic degree

Master of Science

Author: Dennis Schoepf

Presented for the Joint Master Programme on Human-Computer Interaction

Assessed by:

Dr. Bernhard Maurer

Salzburg, 6th of September 2021

Declaration on oath

I, Dennis Schoepf, born on 21.04.1994 in Fürstenfeldbruck, hereby certify that I have adhered to the principles of scientific work to the best of my knowledge and belief and that this Master's thesis was written by me independently. I have not used any sources and aids other than those indicated. I affirm that I have not previously submitted the Master's thesis as an examination thesis in any form, either in Austria or abroad, and that this thesis is consistent with the thesis submitted to the assessors.

Salzburg, on 6th of September 2021

Dennis Schoepf

Personal Identification Number

Zusammenfassung

Dies ist die deutsche Zusammenfassung mit ca. 200 Wörtern.

Abstract

This is an english abstract of the master's thesis with approximately 200 words.

Supplementary Material

Throughout the research of this thesis supplementary material was generated. This includes source code for a web application built for the study and the raw data generated by the participants, as they have gone through the study. This supplementary material can be found within a git repository on a Gitlab instance that the University of Applied Sciences Salzburg provides. Whenever a file or directory of this repository is referred to (usually in the footnotes of the text), you can access the respective file or directory by prefixing the file or folder path with the URL plus the following text: `-/blob/master/`. For more information on the structure of the repository, refer to its `README.md` file. The link to directly access the repository is as follows:

`https://gitlab.mediacube.at/fhs45214/master-thesis/`

Contents

1	Introduction	7
1.1	Motivation	7
1.2	Research Questions & Contribution	7
2	Related Work & Theoretical Grounding	9
2.1	Play	9
2.1.1	The Origins of Research into Play	10
2.1.2	Other Perspectives on Play	14
2.1.3	A Contemporary View on Play	19
2.1.4	Gamification vs. Play & Playfulness	23
2.2	Onboarding in Software Development	26
2.3	Combining Play & Software Development	30
2.4	Design Implications & Attributes of Play	35
3	Methodology	40
3.1	Methodological Considerations	40
3.2	Research through Design	42
3.3	Probing in a Digital Context	46
3.4	Data Analysis	51
4	Research Design	55
4.1	Research Ethics	55
4.2	From Theory to Probe	56
4.3	Probe Implementation	59
4.3.1	The User-Facing Implementation	62
4.3.2	Game World Generation	68
4.4	Data Collection	71
4.5	Participant Selection	74
4.6	The Final Technology Probe	75
5	Results	77
5.1	Playful Design Guidelines	77

5.2 Study Results	78
6 Discussion	90
6.1 Reflection on the Approach	90
6.2 Related Work	90
6.3 Relevancy & Future Implications	90
7 Conclusion	92
7.1 Contribution	92
7.2 Outlook	92
List of Figures	93
List of Tables	94
List of Abbreviations	95
Bibliography	96
Appendices	103
A Data Protection & Consent Form	103
B Probe Development Process	106
C Probe Execution Process	107
D Probe Information Message Types	108
E Probe Data Collection – Questions	109
F Probe Data Collection – Logging	111
G Technology Probe Scenes	112
H Participants	117
I Aggregated Data from the Technology Probe Logs	118
J Qualitative Data – Coding Units	120

1 Introduction

Note for feedback **This is going to be in this part:**

Introducing what will be researched and the research fields involved.

1.1 Motivation

Note for feedback **This is going to be in this part:**

This is taken from the expose and is going to be reworked during the course of writing the thesis

With a projected growth of the number of software developers by about 4.8 million to 28.7 million in 2024 [1] it is paramount to research how these new developers can be onboarded in an ever-increasing number of software development projects. With an increase in the number of tools available as well, like packages and micro-packages in many ecosystems (e.g. Javascript [2]). As many of these packages can be freely combined in development projects, the complexity of these projects increases as well. Therefore it is of utmost importance to ease the onboarding of the increasing number of developers, within the increasing number of software projects with the increased complexity. Improvements in this area could reap real benefits like a decrease in onboarding time or increase in developer productivity and morale can directly affect the success of the respective of projects. While there are tools to help with that, as I am going to lay out later, these often require to already be within the project's code and find your way from a specific point in code rather than helping holistic onboarding in projects. This is especially important for nascent and new software developers that do not know their way around a specific codebase but rather have to explore the project as a whole. I believe that theories, systems, and design patterns from Play Theory can help with that. This area of research is often used to create an enjoyable experience and provides methods for users to let them explore environments by themselves. Miguel Sicart for example states that play is a way of exploring the world [3, p. 3] and this could be applicable on a smaller project-level as well. Therefore using these approaches could potentially lead to a more effective, more enjoyable way of onboarding software developers. This is why my master's thesis is going to be all about: "Playful exploration in Software Development Projects"

Special attention will be given to the emotional side of software development ([4, 5, 6]) and if and how Play could lead to emotional benefits and a more enjoyable onboarding experience.

1.2 Research Questions & Contribution

Note for feedback **This is going to be in this part:**

These are the research questions as of now:

- **RQ1:** How can findings from play theory be used to in developer onboarding contexts in software projects?
- **RQ2:** Which playful exploration strategies, patterns, systems, and theories could lend themselves well to evolve software development projects into a space that enables play?
- **RQ3:** What are the technical, social and personal intricacies of an onboarding context in software development and how do they influence the onboarding experience?
- **RQ4:** How can the pre-discovered strategies, patterns, systems and & theories be applied in actual onboarding settings?
-
- **RQ5:** How do different kinds of software developers experience playful aspects within an onboarding context?

The contribution of this thesis is going to consist of two parts where one of those builds upon the other. Firstly, empirical, qualitative data is going to be generated during problem-centered interviews with developers. This dataset is going to be analyzed and together with the literature research part a list of possible informed implications and insights for the creation of an artifact is going to be put together. Finally, a digital artifact (comparably to [7]) is going to be created as the other main contribution. It is going to be build upon what was researched before and either tries to create a coherent solution informed by the research results or consist of multiple small interactive prototypes each prototyping a single insight. This is going to be decided after gathering the list of implications.

Additional or alternative contribution are recommendations for using play in the investigated context, what to look for, what to investigate, ...

2 Related Work & Theoretical Grounding

The first step towards answering the questions mentioned before, as well as ultimately contributing to the research field, is a thorough scouting of related work. This allows for both a high-level overview of the state-of-the-art in the field of Play and a deeper insight into comparable research specific to the context at hand.

The structure of said scouting is going to be defined by what results are needed as a foundation for the next steps of research within this thesis. As the overall research topic centers on investigating the intersection between the field of software development and Play, this state-of-the-art investigation has to reflect that. This means that there is a need for looking for related work in two different areas of research. One of those two is Play, where it is paramount to gain an understanding of what Play entices, how it can be defined and how it can be leveraged and enabled in different contexts. One of these contexts is the second area of research being investigated in this thesis: Software development in general and onboarding onto software development projects specifically. This then serves as the theoretical grounding for all further work. To gain a more complete understanding of the intersection of Play and Software Development one additional step is required though: Analyzing the state of Play in software development, where it is used, how and to which effect. Ultimately this results in a three-fold research into related work:

1. Research into the state-of-the-art of play as a research field
2. Investigation into the context of onboarding in software development projects
3. Discovery of existing attributes of play in software development

As mentioned these three parts serve as the theoretical foundation for the playful design guidelines, as well as the study later on. Therefore, special attention is given to research that informs both of those as well. This could mean, discovering design implications that are mentioned in the given literature or theoretical grounding for the preparation and the study itself.

2.1 Play

The concept that underlies all that is following in this thesis is the concept of Play. But what is it really, how could you define Play, what does it entice, and how would

you approach researching it? As we are going to find out these are not the most straight-forward questions to answer. Before even trying to answer these questions, it is paramount to clarify why Play would even be a good candidate for this thesis' research. Play in itself is ever present and most importantly inherent to humans. This is displayed quite well in children, where e.g., Fenson & Schell describes children exploring their world through playing, without having explicitly learned to do it: 'It is largely through their playful transactions with people and objects that they gain information about physical and social aspects of their environment' [8]. Baldwin et al. as well have found that children as young as 9 months were able to draw simple inferences about non-obvious object properties after brief exploration phases with those objects [9].

2.1.1 The Origins of Research into Play

While this not necessarily fully translates to other contexts – *Software Development Onboarding* in this case – and there might be great differences in how children play vs. how adults play, it shows that play is innate to humans. There are important distinctions in how that manifests though, which is going to be a further point of discussion in this chapter. To get to that point of discussion play has to be defined in some way. As already mentioned there is no single, concise definition of what Play is in research. Its meaning evolved over time and is subject to influences from the respective paradigms and worldviews behind its different definitions. Independently of these different definitions, there is a need to define a starting point to discuss the meaning of Play.

Fortunately, this starting point is very well-defined in the area of Play. Johan Huizinga marked said point clearly with his book *Homo Ludens: A Study of the Play-Element in Culture* [10]. Within it, he argues in great detail about characteristics of Play, where the differences lie regarding other human behaviors and grounds it within a plethora of cultural examples throughout history. His main contribution that influenced all further work and still acts a point of discussion are five characteristics attributed to Play. While these are not without their fair share of critique as we are going to find out later, they certainly are important to mention and reflect on.

Before diving deeper into these characteristics it is important to note that Huizinga himself constricted his definition to the relation of culture and play: 'Since our theme

is the relation of play to culture we need not enter into all the possible forms of play but can restrict ourselves to its social manifestations' [10, p. 7]. Therefore, he excludes 'the more primitive play of infants and young animals' [10, p. 7] and limits himself on 'contests and races, of performances and exhibitions, of dancing and music, pageants, masquerades, and tournaments' [10, p. 7]. This is something that has to be kept in mind in order to reflect upon his theory as we go further into a discussion of Play at large. Going back to the actual characteristics though, the first one is described as follows:

Here, then, we have the first main characteristic of play: that it is free, is in fact freedom.

Johan Huizinga, [10, p. 8]

Naturally the first follow-up question to this statement is the definition of freedom or free in this context. What does he mean with play being free or freedom itself? Gladly this is clearly described in his work as well: 'First and foremost [...] all play is a voluntarily activity Play to order is no longer play: it could be at best a forcible imitation of it' [10, p. 7] and even though it is mentioned that play by instinct (as e.g. children play) is not necessarily voluntary play, the enjoyment can be freedom. There are two 'sub-characteristics' as well that are notable here: "Play is superfluous" and "it is done at leisure, during "free time"" [10, p. 8]. Overall, Huizinga clearly characterizes Play as something that can not be imposed upon someone but rather intrinsically engaged with. Moving onto the second characteristic, Huizinga writes:

[...] play is not "ordinary" or "real" life. It is rather a stepping out of "real" life into a temporary sphere of activity with a disposition all of its own

Johan Huizinga, [10, p. 8]

What Huizinga illustrates here is that Play and playful activities allow the player to experience a space disconnected from "real" life and therefore fully dive into the Play context. This is closely connected to the third characteristic mentioned by Huizinga:

Play is distinct from "ordinary" life both as to locality and duration. [...] It is "played out" within certain limits of time and place. It contains its own course and meaning

Johan Huizinga, [10, p. 9]

These two characteristics specifically were greatly influential in later research. They marked the first advance into the idea of a *Magic Circle* which is: ‘In a very basic sense, the magic circle of a game is where the game takes place. To play a game means entering into a magic circle, or perhaps creating one as a game begins’ [11, p. 95]. While there is a lot of criticism – Consalvo for example mentions that players always bring outside knowledge about games and gameplay into gaming situations [12, p. 415] and Copier proposes to go beyond the magic circle and rather view games from a network perspective [13, p. 11] – it still holds relevance for discussing the different contexts of play and the real or ordinary. The fourth characteristic brought forth by Huizinga moves onto what is created when playing:

it creates order, is order. Into an imperfect world and into the confusion of life it brings a temporary, a limited perfection.

Johan Huizinga, [10, p. 10]

Positively framed that means: ‘Play casts a spell over us; it is "enchanted", "captivating"’ [10, p. 10], that the order it creates within itself leads to a certain kind of immersion. On the other hand, as Huizinga points out: ‘The least deviation from it "spoils the game", robs of its character and makes it worthless’ [10, p. 10]. That kind of interpretation of Play also leads to Huizinga’s emphasis on the rules of play. He writes that ‘All play has its rules’ and that: ‘The rules of a game are absolutely binding and allow no doubt [...] as soon as the rules are transgressed the whole play-world collapses’ [10, p. 11]. This clearly is a very black-and-white perspective on rules of play and its order. Other approaches to definitions of Play put this into perspective as we are going to find out, inspecting those. Ultimately though, Huizinga argues on the importance of a certain rule set, which definitely can be a prerequisite for Play itself. How important these actually are in the context of this thesis, is going to be subject of discussion throughout this chapter. Lastly, the fifth characteristic described by Huizinga is:

It is an activity connected with no material interest, and no profit can be gained from it

Johan Huizinga, [10, p. 13]

This certainly can be true for the act of playing itself, with the distinction that around Play there absolutely can be material interest involved with profit gained from it –

with the simplest example being the sports industry as a whole. For the remainder of this thesis, this is not going to be a major discussion point, therefore it is just mentioned for completeness.

After laying out in detail the starting point of Play as a research field, it is extremely important to reflect on how these perspectives changed over time. Only by doing that a coherent picture of what makes Play can be painted and later built upon. As mentioned before several points of critique emerged over time, some of which are going to be reflected on in the following.

In order to do this Huizinga's work (as well as its critique) has to be set into historical perspective. Definitions of Play before often described 'play as a tool for the satisfaction of a biological or social need' [14]. These *function-centered theories* also assume that 'playing could in theory have been replaced by some other behavioral technique capable of fulfilling the same function' [14] and therefore 'any function-centered theory necessarily fails to explain why people play' [14]. These considerations ultimately lead Huizinga to fully reject *function-centered theories* and use comparative description rather than quantitative methods to argue in favor of his definition of Play [14]. While this is a very important counter-argument to make, disregarding the functional side of Play goes too far. In order to gain a holistic grasp on what Play is, both the intrinsic, ludic experience and the extrinsic social functions, biological needs and functional means have to be taken into account. This especially rings true in the context of this thesis, where the intrinsic desire to playfully explore is combined with the functional goal of onboarding onto software development projects. Ultimately a broader perspective on Play is needed to fully uncover how to enable and leverage it.

There are other points of critique as well. Hector Rodriguez mentions this in regard to the *irrationality* of Play: 'Huizinga also contents that playing is in some sense and "irrational" activity. Taken at face value, however, this assertion is patently false. Many games depend on strategic thinking and other forms of logical thought' [14]. As discovered earlier Huizinga emphasizes the undoubted rules of a game and the clear boundaries are essential to Play. Rodriguez counters that with a reference to modern art, specifically as done by Kaprow, where he performed without knowing if what he was doing could be considered as art or not [15]. Rodriguez even goes as far as

proposing to embrace this uncertainty as a generative source and to ‘begin designing frameworks for actions that may or may not be considered playful’ [14]. Even with this critique it has to be mentioned that you can read Huizinga also as an approximation rather than an advancement of rigid definitions [14].

What can be criticized as well (although it is not a central part of this thesis) is Huizinga’s: ‘apparent blindness to the importance of politics, which they regard as particularly indefensible, considering the troubled times in which he lived and wrote’ [16, p. 84]. Although, it has to be mentioned that aside from *Homo Ludens*, Huizinga was well aware of the importance of politics in his other work [16, p. 85].

A hugely important point though, especially in regard to this thesis, is the connection of serious and playful activities. Here, Huizinga shows a certain kind of ambiguity through arguing that play does *not* exclude seriousness, while simultaneously painting a sharp line between the two categories of play and seriousness [16, p. 87]. Ultimately it can be concluded that there are is valid critique due to misconceptions, and some ambiguity on Huizinga’s part. Therefore, it is necessary to advance into other perspectives on Play and reflect on other stances towards defining and describing Play.

2.1.2 Other Perspectives on Play

Due to the shortcomings or sometimes simply ambiguous arguments of the preceding authors more perspectives on Play emerged, with especially notable ones brought forth by Ian Bogost [17], Bernard Suits [18] and Brian Sutton-Smith [19]. In the following the key points of their works is going to be laid out and reflected upon, starting with Suits’ work, specifically *The Grasshopper: Games, Life and Utopia* [18].

In this work Suits tries to define what games and playing games means. To arrive at such a definition he creates a fictional *socratic dialogue* (see e.g., [20]) where he dialectically argues to arrive at a series of definitions. In essence the definition of *playing a game* for Suits is:

To play a game is to attempt to achieve a specific state of affairs [prelusive goal], using only mean permitted by rules [lusory means], where those rules prohibit use of more efficient in favour of less efficient means [constitutive rules], and where the rules are accepted just because they make possible such activity [lusory attitude] [...] Playing a game is the voluntary attempt to overcome unnecessary obstacles.

Bernard Suits, [18, p.35]

The definition itself is widely cited, but the actual dialogue behind is often disregarded as Mitchell lays out – specifically mentioning the ‘productive ambiguities of the text, particularly with regard to the relationship between games and society’ [21]. And rather than arriving at ‘apparently universal truths [...] it situates these truths in social context. The text is therefore useful for anyone concerned with the social or political dimensions of games’ [21]. While this is not the focus of the research within this thesis, these definitions are important to mention due to their relevance in the field. And although there is criticism on this definition specifically, e.g., by Back, who writes:

His definition of game playing has some obvious flaws. For instance, it decrees that a game cannot use the most efficient means to win. But why cannot we have a race in this way that uses the fastest means of transport and even allows disabling other racers

Allan Back, [22, p. 5]

The definition still holds value and serves as a point of discussion for future scholars as we are going to find out especially when investigating Sicart’s work. For this thesis specifically there are some aspects worth elaborating on in order to inform the study down the line. The prelusory goal is something that should be included within the practical part of the study. This prelusory goal marks the end of this respective part of the study and has to be defined beforehand. This goal, as Suits describes it, should be achieved by a rule set (lusory means & constitutive rules), which also has to be defined before the implementation of the practical study part. Within the context of this study constitutive rules play a special role. It could be argued that the most efficient software development onboarding (at least on a source code level) consists of interacting directly with said source code. Therefore, the rule set arguably could hinder onboarding on that level, which is why that is a topic of interest going into the practical study. Overall though, Suits definitions was expanded upon greatly and the parts that still bear much value today are rather ambiguous and focused on the intersection of society and games. Thus, it is a vital part of the history of research into play but less vital for the informed implementation of the subsequent study. A perspective on Play that might be able to serve as a foundation for such an endeavor that chronologically followed Suits is Brian Sutton-Smiths body of work, especially *The Ambiguity of Play* [19].

Staying within the theme of an ambiguous approximation of defining Play, Sutton-Smith tries to approach the topic by dissecting ‘the rhetorics that are marginal to play [19, p. VII]’. In this context *rhetorics* can be defined as ‘being a persuasive discourse [...] to persuade others of the veracity and worthwhileness of their beliefs’ [23, p. 213]. The work itself consists of the explanation of seven of different rhetorics, described hereafter, as interpreted and condensed by Wein, as ‘the hypothesis presented here is complicated and abstruse, the proof complex and convoluted.’ [23]:

1. ‘**Play as progress**, in which it is believed that "animals and children, but not adults adapt and develop through play"' ([19, p. 8] as cited by [23, p. 213])
2. ‘**Play as fate**, focusing on games of chance which is in total contrast to play as progress’ [23, p. 213]
3. ‘**Play as power** [...] is a way to represent conflict and to reinforce status’ [23, p. 213]
4. ‘**Play as identity**: "usually applied to traditional and community celebrations and festivals’ [19, p. 10]" [23, p. 213]
5. ‘**Play as the imaginary** including all improvisations and literature’ [23, p. 213]
6. ‘**Play applied to the self**, in which play is solely as a source of gratification or escape for its individual participants’ [23, p. 213]
7. ‘**Play as frivolous**, which focuses on the foolery of play, its undermining of the work ethic, and its reflexivity’ [23, p. 213]

Upon close inspection there are a lot of important layers to unfold within each of those rhetorics. Concerning the context at hand though, there is little intersection and some contradiction on the utilization of a play setting within a professional software development environment. Especially the statement that adults do not develop through Play explicitly contradicts the notion of a playful learning experience in an onboarding process. Additionally, an undermining of the work ethic is not something research in this thesis strives for, arguably rather the contrary. And while the other rhetoric approaches explore important aspects of play and culture and the self, these are less applicable to the research fields within this thesis. Due to that and the inherent ambiguity of the field it is hard to derive informed design decisions from this work. Similar to Suits it rather gives more nuance to an approximated definition of Play. This is only partly helpful for the research goals that should be achieved. On a

positive note though, especially Miguel Sicart referenced and expanded a lot on Sutton-Smith's work. Before we are able to get into Sicart's work there is one more important contributor to the Play research body that is necessary to mention: Ian Bogost. The two works: *Persuasive Games* and *Play Anything* shaped a more modern understanding of Play and games, especially in conjunction with computing. Starting with *Persuasive Games*, the central concept Bogost introduced was *Procedural Rhetoric*, which is a type of rhetoric tied to how computers are functioning. More specifically it is a combination of the procedurality of computing in general and rhetorics as a mode of persuasion or in Bogost's own words: 'Procedural rhetoric is the practice of persuading through processes in general and computational processes in particular' [17, p. 3]. In his book he argues that video games are uniquely positioned to leverage this power to persuade by presenting examples from three different areas: Politics, Advertising and Learning. Between those, the area of learning is of importance for this thesis. Before elaborating on that area though, there is additional clarification needed on the meaning of procedurality and rhetorics as Bogost defines it. Starting with the term procedural, Murray's definition is brought up by Bogost: 'Murray uses the term procedural to refer to the computer's "defining ability to execute a series of rules." (as cited in [17, p. 4])'. Further he explains: 'Procedural systems generate behaviors based on rule-based models; they are machines capable of producing many outcomes, each conforming to the same overall guidelines' [17, p. 4], which serves exceptionally well as a foundation for a rule set described by most of the preceding researchers in the area. Or as Bogost formulates it: 'Because computers function procedurally, they are particularly adept at representing real or imagined systems that themselves function in some particular way' [17, p. 5]. Rhetoric on the other hand is described as a means of persuasion. Bogost describes the evolution from oral rhetorics of the early greek philosophers to visual rhetoric and finally arrives at the term coined by him. Where this gains relevance for this thesis is in its application in so-called *persuasive games*. These are games that utilize the already addressed procedural rhetorics. Bogost presents multiple different examples of such games [17, p. 46-53] and thoroughly explains how certain game mechanics display aspects of such a rhetoric. An example that is brought up is the game *Tax Avoiders*, where a rather simplistic game mechanic and visual language is used to reach a goal removed from the game itself

which in essence is just a jump-and-run game. In this case the player is trying to avoid enemy sprites representing tax agents and is able to use certain tax-avoidance measures between levels. Here the game ‘mounts an interesting and relatively complex procedural rhetoric about tax avoidance strategies’ [17, p. 52]. Such approaches can be (and are) applied in different areas, as was already touched upon before.

In the area of interest for this thesis – Learning – Bogost acknowledges that it is hard to define if video games can be educational, as the question: *What does educational mean?* is hard to answer definitively [17, p. 233]. He further outlines the two dominating, contemporary views on education influence by a behaviorist worldview on one side and a constructivist on the other side. Bogost elaborates on the deficiencies of both when it comes to video games as educational tools. Behaviorists could argue that games can serve as a tool to learn about real-world models, e.g., playing *Microsoft Flight Simulator* to learn about flying aircraft. This obviously does not mean playing such a game makes you proficient enough in flying a real aircraft [17, p. 238]. Bogost calls this a *simulation gap*, ‘the breach between the game’s procedural representation of a topic and the player’s interpretation of it’ [17, p. 238-239]. But the constructivist approach is not without flaws as well. From its ‘perspective, video games teach abstract principles that service general problem-solving skills and learning values’ [17, p. 239]. This view on the other hand neglects ‘video games’ ability to cultivate higher-order thinking skills’ [17, p. 240]. Bogost’s answer to this dilemma is the idea that:

Videogames do not just offer situated meaning and embodied experiences of real and imagined worlds and relationships; they offer meaning and experiences of particular worlds and particular relationships. The abstract processes that underlie a game may confer general lessons about strategy, mastery, and interconnectedness, but they also remain coupled to a specific topic

Ian Bogost, [17, p. 241]

In essence this means that, while video game knowledge cannot be translated 1:1 to real-world knowledge, it might be able to educate about particular topics – such as software development and onboarding in the case of this thesis (which is going to be subject of discussion going forward). What has to be mentioned though, is that ‘Video games teach biased perspectives on how things work’ [17, p. 260] which is exactly why additional research is needed in order to explore its usage in the context of this thesis.

Moving on from this perspective Bogost also expanded on his research into Play and games with *Play Anything*. Central to it is the idea that ...

TODO: Bogost Play Anything

- Central idea
- Things applicable to our context
- Relevance moving forward

2.1.3 A Contemporary View on Play

Up until this point we have discovered that a single, unified definition of Play and Playfulness does not exist. Arguably this is the only constant throughout the history of research into it. This does not necessarily mean that there are only conflicting answers to the question: *What is Play?*, rather that there is a myriad of different perspectives on how it can be answered, each being valuable to explain a certain aspect of it. That does make it hard to derive actual implications for an informed design. An attempt to at least formalize a way of thinking about Play that is based upon much of what came before, was done by a researcher that was already mentioned before, Miguel Sicart. His work *Play Matters*, was regarded positively upon publishing and is widely cited in the field on a number of different topics: Such as social media [24], the mischief and antagonism on the internet [25] or video games as culture [26] and digital leisure [27]. The broadness of these sources possibly show the applicability of sicart's work for a number of different contexts. The applicability in this thesis' context is subject to discussion hereafter together with the general perspective on Play that Sicart offers. Ultimately the questions that should be answered, regarding his work, are:

- What's the essence of Sicart's definition or explanation of Play?
- Where does Sicart's perspective extend upon or differ from previous research?
- Is this a valid theoretical grounding for the research within this thesis?

Starting with the first of those questions, Sicart himself states that: 'Play, like any other human activity, is highly resistant to formal understanding' [3, p. 2] – Therefore he is trying to understand play and why it matters, rather than formally defining it, a take that seems to be reasonable coming from the historically very diverse – *and ambiguous* – approaches to such a definition. Sicart then starts trying to do just that

by opening with the statement: ‘To play is to be in the world. Playing is a form of understanding what surrounds us, and a way of engaging with others.’ [3, p. 1].

Further, he positions his work as an extension of the canon of Huizingan Play, while simultaneously being *Post-Huizingan* in some points we are going to elaborate on in the subsequent paragraphs. Sicart also draws a sharp line to other perspectives on Play that are focused on games or certain objects/toys in particular:

Instead of deriving an understanding of play from a particular object or activity, like war, ritual, or games, I see play as a portable tool for being. It is not tied to objects but brought by people to the complex interrelations with and between things that form daily life.

Miguel Sicart, [3, p. 2]

If its meaning cannot be derived that way, another approach is necessary to explain why play matters and what it entails. Similar to how Sutton-Smith and Huizinga structured their attempt on explaining play, Sicart goes on to describe certain aspects of play. Starting with how ‘**Play is contextual**’ [3, p. 6] and how ‘Play happens in a tangled world of people’ [3, p. 6], with context defined as ‘the network of things, people, and places needed for play to take place’ [3, p. 7]. What is also mentioned in the course of this statement, is the idea of spaces of play and playgrounds. These are vital concepts within Sicart’s book and hugely important for the research design that follows later on and therefore explained in more detail at that point.

The second aspect of play that is mentioned, is that ‘**Play is disruptive** [...] When it takes over the context in which play take place, it breaks the state of affairs’ [3, p. 14]. This allows for different outcomes in situations, from disrupting a context for ‘simple’ enjoyment to going beyond that and by appropriation revealing ‘the inner workings of the context we inhabit’ [3, p. 15]. This claim is of great importance regarding the context under investigation within this thesis. It directly relates to what we want to achieve by combining the field of play with software development: Revealing the inner works of software development projects respectively the surrounding context.

Another aspect of play proposed by Sicart is the idea that ‘**Play is autotelic**—an activity with its own marked duration and spaces and its own conditions for ending’ [3, p. 16]. This might relate to previous research in this field: e.g., the already addressed *magic circle* wherein play happens and the clearly defined rule sets seemingly necessary

for play. Sicart distances himself from such clear-cut boundaries, though and writes ‘the boundaries of autotelic play are not formally rigid; there is no clear demarcation between the world of game and the world at large’ [3, p. 16]. Here we can observe a connection to what Bogost stated concerning video games as educational tools. While the *game world* is markedly different from the *world at large*, certain concepts and systems do translate to this *game world*. Sicart extends on this idea in this case, by verbalizing the ability of play to blur the lines between those worlds:

We play by negotiating the purposes of play, how far we want to extend the influences of the play activity, and how much we play for the purpose of playing or for the purpose of personal expression

Miguel Sicart, [3, p. 16]

Important to note here is Sicart’s addition that this negotiation constantly happens and the purpose of play can change midway through activities. Because of that it is even more relevant to understand in which context play happens and how such spaces can appropriate play.

Sicart further argues that: ‘**Play is creative**, in that it affords players different degrees of expression inherent in the play activity itself’ [3, p. 17], it is ‘the act of creatively engaging with the world, with technologies, contexts, and objects, from games to toys to playgrounds, exploring them through ludic interaction’ [3, p. 17]. Within these statements there is more potential regarding what this thesis is trying to achieve. Especially play as a way to engage and more importantly explore the spaces where it happens, could have the ability to help players (or developers for that matter) cheerfully discover relations in software projects. This closely aligns to Sicart’s conclusion of what play can mean in such a context:

Play is finding expression; it is letting us understand the world and, through that understanding, challenging the establishment, leading for knowledge, and creating new ties or breaking old ones [...] Play is like language—a way of being in the world, of making sense of it

Miguel Sicart, [3, p. 18]

The explanation attempt concludes with the last aspect: ‘Finally, **play is personal** [...] the effects of play are individual’ [3, p. 17]. This already hints at the fact that it is hard to generalize and project findings when researching such a matter. We are going

to refer back to this statement within the research design, as it is essential for the methodological decisions that need answering.

After outlining Sicart’s perspective on play, he continues to go into more detail on certain terminologies surrounding the field of play: *Playfulness* and *Playgrounds/Spaces of Play*. Concerning playfulness, it is crucial to understand the difference between it and play as whole. Sicart writes: ‘The main difference between play and playfulness is that play is an activity, while playfulness is an attitude’ [3, p. 22]. This is an important distinction to make, as it has consequences on the underlying activity of the surrounding context. As playfulness is ‘an attempt to engage with the world in the mode of being of play but not playing’ [3, p. 22], it ‘preserves the purpose of the activity, that it is applied to: it’s a different means to the same end’ [3, p. 26]. What that does mean is that, while you can be in a playful attitude whilst playing, a playful attitude can exist outside the activity of play. Consequently, this allows for different approaches for this thesis’ research goals. Designing for play as an activity or a game in itself is equally possible as putting the original activity in the center but allowing for playfulness to emerge.

Sicart describes the spaces where such activities can happen (or playfulness can appropriate the context of these spaces) by using the metaphor of *playgrounds*. He illustrates them as spaces without any other functionality than to facilitate play – and states that almost any space can become such a playground [3, p. 7]. Further, he proposes two different kinds of playgrounds, *game spaces* and *play spaces*, with game spaces being designed explicitly for game activity and play spaces created to accommodate play but not impose one particular type of play. Sicart illustrates this with an example from the video game world, where game spaces are closed-world game worlds for a particular game and play spaces sandboxes where multiple different play activities can happen [3, p. 51]. A prime example for such a play space this would be GTA Online¹, where the carefully crafted game space is used not only for the intended purposes, quests, and gameplay. There is a huge following that appropriated the space in a different way, in this case *roleplaying* on custom servers (e.g., on the german StateV server ²). Regardless of how players appropriate these spaces, when designing

¹<https://www.rockstargames.com/GTAOnline>, accessed on 10th of August 2021

²<https://statev.de/>, accessed on 10th of August 2021

those spaces it has to be considered what kind of *digital playground* should be created. Both the idea of playgrounds, and playfulness as an attitude lead to implications on how software development onboarding can become a space of play. Before the attributes of play from Sicart’s work as well as the preceding researchers can be condensed into design implications, additional research is needed. It is necessary to dive deeper into the context at hand in order to investigate what kind of research already exists. Thus, Sicart’s work is going to be more closely inspected in regard to context-specific statements later on, when categorizing the attributes of play found throughout all the preceding (and subsequent) literature research.

At this point we should have at least some understanding of what play entails and at least a grasp on how many perspectives on defining play emerged over time. To actually take that information into actions towards answering the research questions, more work is needed. As said, this involves deeper investigation into the context, but also discovering already existing applications of play in software development environments. Before that can happen an important distinction has to be made between a widely used term connected to play, which is the concept of *Gamification*.

2.1.4 Gamification vs. Play & Playfulness

The term gamification is used a lot throughout different industries and can be interpreted in different ways. On the actual definition, there is consensus, though. Gamification can be described as the usage of video game elements in non-video game contexts, mostly to improve user engagement [28]. Where interpretations start to differ, is in what those video game elements can be. Sailer et al. are connecting some widely used game mechanics in ‘gamified’ contexts as visible in *Table 1*. The psychologic effects these mechanics aim to trigger are well researched and documented in *Table 1* as well.

While some of these mechanics were already touched upon in literature on play (e.g., playing in teams, meaningful narratives around play), others were not mentioned explicitly as key game mechanics. To be precise, these are: *Points, Badges, Leaderboards & Performance Graphs*. In the world of gamification, though, these are some of the mechanics proposed most often (e.g., by Zichermann in [38, p. 3] and [39, p. 35-50]). As one can see there is a large difference in perceived importance of what

Game Mechanic	Theory
<i>Points</i> for the successful accomplishment of tasks	Measuring in-game behavior, continuous and immediate feedback and reward [29]
<i>Badges</i> as collectable representations for player’s achievements	Visibly showing accomplishments [30], serving as task goals or as virtual status symbols [31]
<i>Leaderboards</i> ranking players based on their relative accomplishments	Artificial competition as a way to increase engagement of players with similar skill levels [32, 31]
<i>Performance Graphs</i> to show the performance of a particular player in respect to their previous attempts	Triggering <i>mastery orientation</i> , as it is described in motivation theory [33, 34]
<i>Meaningful Stories</i> as a narrative context	Enrich barely stimulating contexts with cooperationmotivating narratives, especially when aligning with a player’s interest [35]
<i>Avatars</i> as visual representations of players	Often chosen or created by a player for identification purposes [31]
<i>Teammates</i> to interact with (non-player characters & other players)	Inducing conflict, competition or cooperation and creation of teams [31, 36]

Table 1: Gamification mechanics and their psychologic effects (condensed by me, based on Sailer et al. [37, p. 373-374])

makes a game vs. a ‘gamified’ experience. Consequently, this is also the most criticized aspect of communication. One of the most prominent critics has already been mentioned, Ian Bogost. In his provocative essay ‘Gamification is Bullshit’ [40], he describes those mechanics as the key to gamification:

In the traditional version of gamification, the process involves the adoption of simple, repeatable, scalable feedback systems such as points, levels, badges, and other rewards

Ian Bogost, [40, p. 68]

He follows this up with what is often criticized about using or propagating these techniques as key game mechanics – the notion that these elements are central to game design [40, p. 68]. In Bogost’s opinion this is not remotely the case, as these techniques can complement game design but do not make a game by themselves. Focussing solely on engagement, he continues, disregards many aspects of what makes games worth playing, like in-game mechanics, aesthetics and carefully crafted worlds. What he also adds, is that for gamification advocates, these aspects maybe are not key, but in his words, the ‘nature of games [for gamification advocates] is whatever is most easily

abstracted, packaged and sold’ [40, p. 68]. Bogost then ultimately concludes with the statement, that gamification as he describes it has little in common with game design and development [40, p. 72].

Overall his essay reads like a scathing condemnation of gamification and its advocates and while there is a lot of very valid criticism in his statements, the argument might not be as black and white. What holds true from his stance is the fact, that a narrow focus on or even the propagation of *Points, Badges, Leaderboards & Performance Graphs* as key mechanics disregards much of what makes games fun and engaging in the first place. Packaging these mechanics as business solutions in environments that sometimes even hinder playful appropriation, further robs games from what makes them stand out. It is important to note that Bogost does not argue against using games in non-gaming-contexts at all, as we already elaborated on in his view towards games as educational tools.

It has to be said, that there certainly are gamification advocates that share some of these thoughts, albeit in a less black-and-white manner. Knaving for example, agrees that ‘common approaches to gamifying activities focus too narrowly on rules and reward systems as a layer separate from the main activity’ [41, p. 134]. Nonetheless, Knaving notes that ‘Gamification can be used to make activities more engaging’ [41, p. 134]. Additionally, Knaving states that:

Gamification has been viewed as a complement to designing for playfulness, but if play is an integral part of games, it is also possible to argue that affordances for playfulness should always be considered when designing gamification

Kristina Knaving, [41, p. 133]

Thus, one could argue that there does not need to be an either-or decision between gamification and designing for play and playfulness. It rather is a matter of the goals a game designer wants to achieve and which mechanics can deliver these goals. It certainly is possible that this is the singular goal of *user engagement* (without assessing how valuable such a narrow goal might be). In my opinion this is too narrow-minded of a goal and while it helps when crafting business solutions, it might even hinder creating spaces of play in non-gaming contexts.

Ultimately the question is what to take from all of this for the remainder of this thesis. As already laid out with the research questions, the goal is not to increase engagement,

but make the onboarding process more playful and enjoyable. Therefore, gamification, in its original, narrow sense is not the right approach for such a goal. This does not mean, that none of the mechanics typical to gamified experiences should be used in a possible solution. Rather that, these mechanics can be utilized as complementary to a playful game experience deployed to a non-gaming context. For this reason some of the initial gamification mechanics are going to be included as possible implementation details. But, because of the nature of the overall contribution goals of this thesis, gamification as a whole is not going to be considered as theoretical grounding.

2.2 Onboarding in Software Development

After exploring both possible theoretical foundations and related work in the field of play, attention has to be given to the context that is going to be investigated – in our case *onboarding processes in software development projects*. In order to discover the intricacies, challenges and approaches of such onboarding processes, at first a definition has to be synthesized. Fortunately, Rebecca Yolande Yates did exactly that in her doctoral thesis in the field of Software Engineering [42]. This synthesized definition of onboarding is as follows:

Onboarding is the process by which existing team members help a newcomer to acclimatise to working on their team. Newcomers must take in a great deal of social and technical knowledge; for software developers, this typically includes comprehension of unfamiliar code

Rebecca Yolande Yates, [42, p. 33]

This process consists of many parts on different levels. These parts can loosely be categorized in organizational, technical and social aspects [42, p. 33-35].

Organizational onboarding ‘is a process through which new employees move from being organizational outsiders to becoming organizational insiders’ [43], where a newcomer gets to know organizational intricacies and structures. The technical aspect of onboarding is more concerned with the already mentioned comprehension of code and program comprehension. Lastly the social aspect consists of the many interactions within a team of software developers (and other project members). Yates also identified three different schools of literature on onboarding [42, p. 39-40]:

1. **Psychology-Centered Approaches** – Focusing on program comprehension and programming skills
2. **Process-Centered Approaches** – Investigating step-by-step approaches of developers (*Information Seeking, Concept & Feature Location*)
3. **Developer-Centered Approaches** – Focusing on social processes, tools and environments

These approaches differ not only in what they are centered around but also in how successful onboarding can be achieved. Depending on project specifics – e.g., the social structure of teams, the tooling, the skills of the involved developers – as well as the goal of the respective onboarding process, certain approaches might yield better results than others. These approaches also encompass a variety of different methods described in literature that can be further categorized. Yates mentions two of these categories specifically, *Push* and *Pull* ([44] as cited in [42, p. 29]). Push in this case describing the information pushed towards the newcomers, pull on the other hand when newcomers request information by themselves.

How onboarding should be structured and how it can be supported, also depends on the specific challenges of individual projects. Within onboarding processes in software development these can be numerous. Dagenais et al., for example points out that obstacles like documentation, that is lacking necessary information, or tooling issues hinder onboarding progress for newcomers [45]. On a more technical level, the understanding of the source code and the programming languages used also matter [46]. The ease of onboarding is also more directly impacted by the size of the codebase and the tools used in the onboarding process [47]. In providing the correct information for each individual newcomer also lies some difficulty, as these have very different information needs to developers experienced in a project [48]. Yates also adds more possible newcomer issues [42, p. 35]. These are, in no particular order, worries about their own performance ([46] as cited in [42]), a project’s toolkit ([49] as cited in [42]) and administrative frustrations ([50] as cited in [42]).

Despite these complexities for businesses as well as open source projects it is worth to invest into onboarding processes. Study results vary on how much time is spent until a newcomer can be considered experienced within a project. Zhou and Mockus investigated reports of developers that were fully productive after six to twelve months,

with onboarding to very specific tasks taking up to three years [51]. DeMarco and Lister on the other hand estimated that each newcomer costs a team approximately three lost work-months ([52] as cited in [42]). So, while the exact amount differs from study to study and project to project, there is great interest to at least somehow reduce the time spent in this stage. Alternatively the goal could also be to at least ease the burden for newcomers to possibly increase employee retention. After clarifying a lot of the intricacies and challenges in onboarding, it is crucial to get a sense of possible approaches from literature.

Dagenais et al. offered the metaphor of a *project landscape* for development projects [45]. They grouped features of a project within groups of *features*, *landscape (orientation) aids* and *obstacles* positioned in such a landscape. Obstacles being the already mentioned documentation and tooling issues for example. These landscape features were then assigned to the different aspects and stages of a project, like product features, processes, team, documentation and organizational context [45, p. 278]. Orientation aids included IDE tools and walkthroughs but also social methods, like team meetings and code reviews. Obstacles on the contrary included upfront courses, proprietary tools or inadequate documentation [45, p. 278]. Dagenais et al. found that experienced *landscape inhabitants* (= *other project members*) are key to onboarding success. They also propose the creation of ‘micro and macro views that show the relationships between different landscape features and routes that can be taken’ [45, p. 284].

A more recent example of a technology-aided approach to support newcomers in their onboarding journey comes from Dominic et al. [53]. They propose a use of voice recognition bot to deliver personalized answers about certain questions newcomers might have. More specifically a stage and control flow diagram is presented for possible implementation of such a system. Within this diagram multiple possibly helpful steps are included e.g., automatic fetching of Stackoverflow Data, automated selection of relevant issues and recommendation of experienced contributors as contacts [53, p. 48]. A compelling argument is made on how this approach could help in onboarding, but the authors themselves note a few limitations [53, p. 49]. First and foremost an extensive data set is needed to train a conversational bot. At the time of publish this data set only included 20 existing conversations, as the data collection process is very

time and cost intensive. In addition to that integration of voice recognition adds another layer of technical complexity with unsure levels of success. Overall such a solution is not expected to land anytime soon.

Lastly, Steinmacher et al. proposed guidelines for open source projects specifically on how to onboard new collaborators. They divided these guidelines into three areas, *Contribution Process, Social Behavior & Technical Guidelines*. Within the contribution process, newcomer-specific portals, identification and dismissal of outdated information, pointing newcomers to easy issues, and an up-to-date issue list are presented as guidelines [54, p. 7-8]. For social behavior, answering quickly, kindness to newcomers and, identification of possible mentors is mentioned. And finally, on a technical level, an easy and well-documented local build of a system and a well-documented code structure are deemed valuable.

Before concluding this chapter it is important to note, that onboarding can differ quite distinctively in different project contexts. Open source projects for example have different challenges to proprietary business projects. Another hugely important distinction to be made is if the project team is colocated or works remotely. While some of the literature presented might be applicable for both open source projects and business projects (e.g. [45]), others are very much geared towards remote open source projects (e.g. [54]). These open source projects have unique requirements with less direct social contact, more casual contributors and mostly voluntary work [54].

Concerning remote onboarding contexts Yates also mentions additional difficulties in the onboarding process, like ‘social team integration, environment setup and unreliable videoconferencing facilities’ [42, p. 35]. Depending on the work environment of the participants of the following study, exploratory data could be gained on either of those settings. To address the differences in context within the study questions on the participants’ professional environment are going to be included.

To conclude this research into foundational and overall related work on software development onboarding, its impact for the next steps of research has to be laid out. Firstly, some guidelines and other discovered study results are going to serve as a foundation for a technical implementation. These are going to be mentioned again, with the rationale behind including them, in chapter 2.4. Secondly, a decision has to be made on where the proposed extension of the research body locates in relation to the

related work. This is going to be laid out in detail during the documentation of the research design. In addition to that some approaches and techniques are going to be part of the design implication in chapter 2.4. Lastly, due to Yates' positive conclusion on push-based techniques [42, p. 197] the following study is going to extend upon these findings specifically.

2.3 Combining Play & Software Development

To get an even more complete picture of how play could be a part of software development, it is paramount to research already existing combinations of it. In order to get such an overview in the following sample applications from industry as well as from research are going to be presented. An upfront disclaimer has to be given that game development is not going to be mentioned as a combination of these fields. Even though it is concerned with the creation of games the actual development process does not necessarily have to be as playful as the end result. Nonetheless, the findings of this thesis could be transferred to such a context as much of the same challenges (as in business and open source projects) apply.

The first noteworthy example of a combination of play and software development or rather the text editing part of software development is *Vim Adventures*³. This example is also mentioned by Ian Bogost in one of his works. Bogost explains it as a game to 'learn the intricate keyboard shortcuts in the Vim text editor' [55, p. 68]. *Vim*⁴ (or *Vi* its predecessor) is a text editor with a notoriously steep learning curve, 'to many beginners, vi looks unintuitive and cumbersome [...] in addition, there seem to be so many commands' [56, p. 3]. After this steep learning curve it allows you to achieve complex tasks with just the keyboard and a few keystrokes, which can help hugely in editing productivity. Because of that there is great interest in flattening the learning curve or help newcomers to the editor in other ways. Even into the editor itself an interactive tutorial is integrated [57]. *Vim Adventures* approaches this from a playful angle. A carefully crafted game world is used as the setting for different characters and game objects that can be interacted with, as shown in *Figure 1*.

The value as a teaching tool lies inside the navigation commands for the player

³<https://vim-adventures.com/>, accessed on 14th of August, 2021

⁴<https://www.vim.org/>, accessed on 14th of August 2021

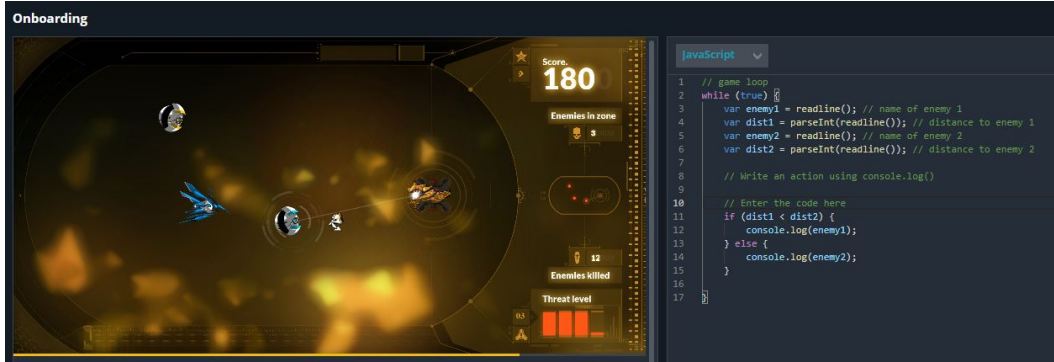
Figure 1: *Vim Adventures* Game Scene

character. The controls for said navigation exactly match the navigation commands within the *Vim* editor. While the player learns these commands (e.g., h,j,k,l keys for directional navigation as also shown in Figure 1) for in-game navigation these commands can simultaneously be used in the text editor. This allows for playful familiarization, possibly motivating the player/user to keep learning despite the steep learning curve of the editor itself. In conclusion this example shows how a carefully crafted game can be able to provide educational value in a non-gaming context. Thus, this approach could act as a blueprint for a possible implementation of a technology-aided onboarding solution.

Another approach was taken by *CodinGame*⁵. *CodinGame* is a mixture between an Integrated Development Environment (IDE) and a game and aims to educate developers on different programming languages and programming concepts. Within the *CodinGame* environment, there are different tasks given to the player. These revolve around the visualization of the player character, game objects and enemies on a part of the screen, as can be seen in *Figure 2*.

Players then can tackle these tasks by writing valid source code in popular programming languages such as Javascript, C#, C++, Java and more. Within the source code it is possible to reference the game objects entering the visualization on the other part of the screen. Through writing game-specific code you then are able to interact with these game objects to kill the enemy objects for example. After the

⁵<https://www.codingame.com/>, accessed on 14th of August, 2021

Figure 2: *CodinGame* Game Scene

player has written the necessary code, it is possible to re-run the respective game scene and thus see if the task has been successfully completed. Contrary to the previous example, *CodinGame* does not try to teach concepts by matching mechanics to an outside concept. It rather uses writing source code as the central concept and only visualizes software development tasks playfully.

The *Starbase* Game ⁶ represents another possible approach. *Starbase* is a massive multiplayer space adventure game. Unique to it, is its very detailed and intricate universe with a lot of different game mechanics and components to interact with. Instead of putting software development at the center or matching game mechanics to tool commands, *Starbase* leverages software development in another way. Programming is used here as one of many game mechanics throughout the game. It allows for sophisticated control of in-game electrical devices. Code can be written in a custom programming language created specifically for the game. With that language nearly every technological device within the game universe can be controlled. To summarize, programming or learning commands connected to programming is not a goal in itself in this case. Instead, it is used just as a single mechanic within a game, where it allows for extremely fine-grained control over in-game components, which could be hard to implement visually.

Besides these sample applications of play from the industry there are also similar attempts. One of them very recently by Lopez et al., where the usage of LEGO® Serious Play as an education tool for software development workflows was investigated [58]. The study was setup in a way that students got handed a process definition to follow that matched popular project management techniques in software development. (e.g., a

⁶<https://starbasegame.com/>, accessed on 14th of August 2021

waterfall-style process and an agile process). Students then had to adhere to these processes with the overall goal to create a chair built out of LEGO® bricks. Depending on the given project management technique the process consisted of requirements, analysis, implementation, design, and evaluation phases in different orders. After students went through these phases, lecturers gave more context on these techniques and explained common pitfalls and advantages. Lopez et al. reported positive results within this study with students largely agreeing that it was highly fun and motivating to go through such processes playfully. The team environment together with lecturers acting as customers, allowed the students to hone their soft skills as well, as they reported. While this study does not resemble a game, it shows how toys can be appropriated into non-play contexts. This in turn allows for a playful learning experience, possibly increasing learning success.

Overall it can be said, that there is a great number of different approaches to utilizing the power of play in software development (or vice-versa, as shown with *Starbase*) with different benefits gained from it. Within the implementation phase of the study, these different approaches could serve as possible implementation options. In essence these options are:

- Leveraging commands or tools used in software development as key game mechanics – similar to *Vim Adventures*
- Using source code created by the player as the main way of interaction within a game – similar to *CodinGame*
- Using programming as one of many mechanics in a game – similar to *Starbase*
- Using toys or other elements of play to visualize or simulate non-gaming concepts – similar to Lopez et al.’s approach

Concerning specifically using elements of play in the onboarding process of software development there is a clear lack of research, though. This reinforces the need for exploratory work into this exact topic, which ultimately is the goal of this thesis.

Lastly, after presenting these different approaches, there is one additional aspect of the software development context to clarify. The general openness towards integrating playful elements in such a process. A generalizable answer is hard to give in this case, given the large number of individual software developers already mentioned in this thesis’ introduction. An attempt at answering the question can be made by

investigating playful events and elements within the software development field. This could allow for an insight into the openness towards play.

One kind of those events are so-called Capture The Flag (CTF) contests. These have their origins in cybersecurity conventions. An example of such a contest would be the DefCon CTF⁷. Within such contests, teams of developers compete against each other by trying to compromise the other team's server. More specifically, teams are required 'to leverage their knowledge of vulnerability detection' [59, p. 134] in order to attack and defend a virtualized host server. Points were then rewarded for both securing a team's own services but also compromising the enemy team's services [59, p. 132-135]. The competition as well as the special rule sets teams agree on definitely resemble elements of play that were already mentioned.

Another kind of event rising in popularity are so-called *Hackathons*. These can be defined as:

an event in which computer programmers and others involved in software development collaborate intensively over a short period of time on software projects [...]. These hackathons are encouraging of experimentation and creativity. and can be challenge orientated

Gerard Briscoe, [60, p. 1]

Briscoe also proposes that elements of the hackathons originated from so called Local Area Network (LAN) parties. These are gatherings 'of people with computers or compatible computer game consoles' [60, p. 3]. Therein one can see the playful origins of such events very clearly. But also the hackathons themselves show similar elements of play to the CTF contests mentioned before with rule sets for the teams as well as possible competition. This holds true even more, when looking at *game jams*. These can be defined as 'an accelerated opportunistic game creation event where a game is created in a relatively short timeframe' [61].

Besides those events, there even are signs of playfulness within the daily work of developers. While the coding work itself normally revolves around rather strict programming structures and syntaxes, playful elements can be discovered there as well. Uncovering all of those is out of scope for this thesis, so one exemplary element is going to be presented here. As a short preface it has to be understood how software

⁷<https://defcon.org/html/links/dc-ctf.html>, accessed on 14th of August 2018

developers collaborate on source code. Typically, this is done with a version control system, where multiple developers are able to contribute code to. This is done by aggregating changes in source code, then attaching a message and ultimately pushing the code changes to a place where other developers can access these changes and build upon. The act of aggregating and attaching a message, is called *committing*. This is a central command within version control systems, as the attached messages communicate meta-information on the code changes to team members. This communication is usually held rather short and precise, but even within these commit messages playful elements can be situated. An example for this is *Gitmoji*⁸, a project that aims to include emojis within those commit messages. That usage of emojis, as Hjorth et al. point out:

has been defined as playful (Sicart 2014 [original citation [3]]). This is epitomized by the playfulness of the paralinguistics – emojis ('picture characters'), emoticons (typographic characters), stamps and stickers. [...] The use of paralinguistics has become an increasingly popular and playful way of personalizing digital media communications.

Hjorth et al. [62]

Overall one could assume that at software developers might be open towards playful elements given the field's occasional openness towards ludic features and events. These are not generalizable statements at this point, though. Therefore one of the additional goals of the upcoming study is to gather more data on this statement.

2.4 Design Implications & Attributes of Play

After all the research done into possible theoretical foundations and related work in general, it is crucial to derive a set of attributes of play and possible design implications that inform the next chapters of this thesis. This is especially important for an informed study design and even more so for an informed implementation. Consequently, this is going to be done in the following. Based on the literature that was analyzed beforehand a non-exhaustive set of attributes are going to be derived. For each of these attributes, the theoretical grounding is presented as well. This is going to result in a list of attributes that inform playful design implications, that in

⁸<https://gitmoji.dev>, accessed on 15th of August 2018

turn serve as the foundation for further explorative research as is going to be laid out in chapter 3. The theoretical grounding mentioned, mainly is derived from Miguel Sicart's *Play Matters*, as it extends upon much of what came before in research into play. It also contains numerous actionable implications to explore further upon – in contrast to an also influential but more abstract and philosophical approach like Suits' grasshopper. Where applicable, implications and attributes from other contemporary works are also included.

The first attribute of play is, that **play is a balance between creation and destruction**. This is based upon on what Sicart describes as, 'Play is the struggle between order and chaos, between the will to create and destroy' [3, p. 5]. This statement can very well be transformed into an actual playful feature of an onboarding experience into a software development project. As such, it is a prime candidate to explore further in the study that is following.

Utilize already existing ludic features of a context is the next implication from literature, based on Rodriguez [14]. Some of these ludic features were described in chapter 2.3 and can be built upon during the study.

Rodriguez also proposes to 'embrace conceptual uncertainty as a generative source [...] begin designing frameworks for actions that may or may not be considered playful' [14]. When transferring that to an actual design implication, this can be interpreted as **embrace uncertain outcomes of playful elements**. That arguably is less of a directly actionable implication, rather a consideration to make. Nonetheless it is a noteworthy addition to this list.

Another consideration Sicart mentions is that 'Play not as an activity of consumption but as as an activity of production [...] play is a way of engaging and expressing our being in the world ' [3, p. 5], which means that **play is an activity of production**. Next up is an implication taken from a case study Sicart presented in *Play Matters*. There he describes the case of an application called *The Accidental News Explorer*. It pulls random sources of news to show the user. Sicart describes it as a serendipitous approach, that 'forces us to look at its choices with playful astonishment'. Thus, **utilize serendipitous events** can be included as a design implication.

Another crucial point brought up by Sicart is the idea of 'disrupting the normal state of affairs by being playful [...] in that move, we reveal the inner workings of the context

that we inhabit’ [3, p. 15]. This resonates extremely well with the overall goal of onboarding into software development projects. Within onboarding we want to reveal the inner workings of the project a developer is onboarded to, so there could be huge potential in applying Sicart’s proposal to **disrupt the conventional state of affairs**.

One more implication with a lot of potential based on the context at hand, is described later on by Sicart, the idea that ‘creative appropriations of data reveal new knowledge through play [...] even in the most rigid of context’ [3, p. 28]. This points to playful visualizations of the context as a promising way to increase the context-specific knowledge. Therefore the respective design implication would be, to **visualize creative appropriations of data**.

An additional implication – also presented as a case study by Sicart – is derived from *Newstweek*. *Newstweek* is an application that alters news headlines of major providers in real time, ‘to playfully manipulate news consumption [...] shattering our assumptions on networks, news, and consumption of stories’ [3, p. 29]. **Playfully manipulate the normal state of affairs** is therefore a design implication that can be derived here.

Sicart continues to propose elements of playful designs specifically. He mentions them being ‘by definition ambiguous, self-effacing, and in need for a user who will complete them’ [3, p. 31]. This can be directly translated into an implication of: **Let the user complete intentionally ambiguous designs**. In the course of this argument Sicart also mentions that ‘what, playful design focuses on is the awareness of context as part of the design’ [3, p. 29], so that is something to consider as well: **make the context a part of your playful design**. This also closely aligns to Bogost’s statements on games as educational tools that are able to teach about certain concepts from a specific context.

Create companions with personality, is another design implication taken from Sicart’s work. He specifically mentions this idea, while elaborating on Siri – ‘Siri is a playful design that breaks our expectations and gives personality to software [...] Siri becomes a companion more than a tool’ [3, p. 32].

Before tackling a larger subsection of implications, another noteworthy comment by Sicart is going to be considered. Within said comment, Sicart specifies that ‘playful designs could be described as somewhat successful toy designs, that is, as objects that

by design allow users to toy around with them’ [3, p. 40]. Reconstructing that into a design implication makes for the statement of **think of designs as toys and let users freely play with them.**

Next up, the topic of *procedurality* is mentioned numerous times by Sicart, who partly extends upon Bogost’s work on procedural rhetoric. Therefore, multiple design implications can be inferred specifically on procedural designs. At first though, a short reminder on what those procedural systems are, they ‘generate behaviors based on rule-based models; they are machines capable of producing many outcomes, each conforming to the same overall guidelines’ [17, p. 4]. In no particular order, these are the design implications derived for procedural systems as a whole and procedural toys specifically:

- **Allow the player to both watch the procedure unfold but also alter it** – based on ‘procedural toys [...], are paradoxical objects that put their users in the double role of performer and voyeur [...] We play with them to see how they behave, how they react’ [3, p. 41]
- **Visualize the procedural world’s conditions** – based on ‘Procedural toys are mesmerizing because they are frames of the otherness, because they are tiny worlds that operate by their own condition’ [3, p. 43]
- **Let the player test the limits of the system** – based on ‘Much of the joy in interacting with these procedural toys comes from testing their very propness as we figure out where the seams are and what we can build with them’ [3, p. 57]
- **Let the player explore and test without too much guidance** – based on ‘Software toys like sim City and other procedural toys are also play spaces to a large extent – spaces of possibility created to explore with rules in order to see what happens’ [3, p. 51]

Moving on from procedural systems and toys, there are some more implications obtained from Sicart’s statements. A statement that stood out was, that ‘designing for play means creating a setting rather than a system, a stage rather than a world, a model rather than a puzzle. Whatever is created has to be open, flexible, and malleable’ [3, p. 90]. This also resonates with the statement, that ‘the game designer proposes and deploys an object into the world, letting it speak for itself and be spoken through’ [3, p. 90]. Reconstructing that into an actual implication, would mean **treat**

the design as a deployable, flexible object rather than a polished piece.

In addition to the design implications from literature on play, there were also some specific guidelines on onboarding experiences discovered through the extensive literature review. Similar to the play focused design implications these are laid out in a non-exhaustive list as follows:

- **Recommend specific (easy-to-achieve) tasks to beginners** – based on ‘To promote early exploration, newcomers are often assigned small bug fixing tasks, chosen to introduce them to the code while minimising risks to the rest of the team; this is because maintenance is regarded as requiring less design knowledge than new feature development’ ([63] as cited in [42, p. 36]) and ‘One way to make the path easier is to point newcomers to easy tasks’ [54, p. 8]
- **Recommend experienced mentors to newcomers** – based on ‘experienced developers are key to understanding a project’ [45, p. 284] and ‘Identify mentors or experts’ [54, p. 9]
- **Visualize the micro and macro structure of projects** – based on ‘What about creating useful maps: micro and macro views that show the relationships between different landscape features and routes that can be taken’ [45, p. 284]
- **Document the code structure** as laid out by Steinmacher et al. [54, p. 10]
- **Identify and dismiss outdated information** as laid out by Steinmacher et al. [54, p. 10]

That concludes the list of design implications derived from literature on both the context and research into play and playfulness. These implications can now be used as a foundation for the next step within this thesis. As such, they are going to serve as one of the anchor points for the explorative study design described in the following chapter.

3 Methodology

After laying out the existing research and deriving implications from it, the next step consists of extending these findings. To achieve that a scientific study is in order. How that study is going to be structured is subject of discussion within this chapter. In the following the research questions are going to be revisited as a starting point for that discussion. For each of the unanswered questions adequate scientific methods to answer them, have to be found. To determine these methods in this chapter, at first additional considerations are laid out. After that, the chosen methodological area(s) is (are) described in more detail. The reasoning and final selection of methods are presented subsequently. Finally, the data generated through the chosen methods has to be analyzed. The way in which this is done, is illustrated at the end of this chapter.

3.1 Methodological Considerations

To choose the right methods for the research questions at hand, additional considerations have to be taken into account. These considerations consist of the nature of the question (is it an exploratory question, do we want to measure effectiveness, ...) and the goal of these questions. The first research question (**RQ1**) is, *How and which theories & concepts from research on play can be used in the onboarding process of software development projects and how are developers experiencing them?*. As this is the main research question it is not going to be answered using a single method but rather through collectively answering the other questions and condensing the results. The second research question (**RQ2**) is: *Which playful themes could lend themselves well to evolve software development projects into a spaces of play?*. These strategies, patterns, systems and theories can also be described as implications for those spaces. The foundation for these design implications was discovered in chapter 2.4. That foundation is expanded upon in the following chapter, where from these implications in literature implications for a possible design are formulated. The question of which of the implications are effective in onboarding processes is out of scope for this thesis though, as the focus lies in exploring possible implications to start with. Nonetheless, initial, qualitative feedback on a subset of implications that could be implemented should be gathered.

For the third research question (**RQ3**) – *What are the technical, social and personal intricacies of an onboarding context in software development and how do they influence the onboarding experience?* – much of the research was already done. Similar to RQ2, the intricacies of the process were described through researching related literature on exactly this topic. Additional qualitative data should be gathered to answer RQ2 within the upcoming study. It is not the goal, however, to develop generalizable statements about the process. The overall goal is an exploratory one and therefore possible participant feedback is only used for the argumentation of the qualitative results.

In contrast to RQ2 and RQ3 for the fourth research question (**RQ4**) – *How can the identified themes be applied in actual onboarding settings?*, as well as the fifth research question (**RQ5**) – *How do different kinds of software developers experience playful aspects within an onboarding context?* – there is no existing body of work to rely on. Therefore the upcoming study should aim to provide an answer to those two questions in specific and gather additional data for the aspects mentioned concerning RQ2 and RQ3.

To discover adequate methods, first RQ4 has to be deconstructed to uncover what exactly should be researched. There are two parts of the question that help with this. Firstly, the notion of *applied* and secondly the phrase *in actual onboarding settings*. Applied, strongly points to the practical creation of something that incorporates the implications identified before. A possible approach for this could be to pick a single implication, implement it and then extensively test it with study participants. This could allow for generalizable results for this specific approach, where one could produce reproducible results. While such an approach could deliver valuable insights, the downside is its narrow focus on one implementation detail. Another approach could be to implement and maybe even combine multiple implications in one design and gather data on the participants' subjective experience towards them – which would be the goal of RQ5. The design process of such an approach could also provide data on possible applications for multiple implementations, which in turn allows for a broader answer to RQ4. The downside of researching multiple implementations at the same time is the possible blending of effects of multiple implications. This means that no definitive answer on the effectiveness of one specific implication can be given. On the

other hand it allows for participant feedback on multiple design implications, based on which future research can be done on promising aspects.

As one can see here, there is no ‘silver bullet approach’ here, rather there are considerations to make on which approach provides more of the desired results. For this thesis these are – as already mentioned more than once before – very much focused on taking a step into uncharted territory and explore different paths within that. Thus, exploratory, open-ended results generally what is aimed for here, which makes for an easier weighting of the arguments presented. This means in the following methods are considered, that are able to deliver on this and produce data on a multitude of possible implications.

The second part of RQ4 (*actual onboarding settings*) narrows down options for methods further. As an important aspect of this thesis as a whole is the combination of play with the context – the aforementioned *actual onboarding settings* – investigating design implications within this context is necessary. This shows the need for a method where the implementation of the design implications can be deployed in the context.

A methodological area that could provide this is *Research through Design (RtD)*, where through a design (e.g., the implementation of the identified implications) research can be done. Important to note here, is the need for additional data to answer the explorative and personal aspects of RQ2 through to RQ5 as mentioned earlier. To identify if RtD as a whole and its specific methods in particular can deliver that, a more detailed description is given in the subsequent chapters. The chosen methods are then laid out in particular, as well as the chosen analysis methods.

3.2 Research through Design

To gain a clear picture of what RtD is, it is important to understand the differences between design and research as a whole. Fortunately, Stolterman describes these differences very well. Research, or more specifically scientific research, he explains, drives towards the existing and to universal knowledge. Specifically Stolterman mentions that, ‘The aim of science is to formulate universal knowledge that explains the complexities of reality on a level removed from specifics and particulars’ [64, p. 57]. On the other hand, ‘design deals with the specific, intentional and non-existing’ [64, p. 59]. Certainly this kind of research has proven very useful to generate ‘knowledge

about natural objects and phenomena’ [65, p. 3]. Research on and with man-made *artificial* objects asks for a different science, as Simon has argued even back in the late 1960s [65]. Simon also further defined aspects of such a science of design, describing designers as everyone who ‘devises courses of action aimed at changing existing situations into preferred ones’ [65, p. 111]. This notion of creating preferred futures generally prevailed. Coughlan et al. specifically mentioned designing prototypes as a way to create such futures and contrasts it to the traditional science approach:

In its essence as a design synthesis tool, prototyping represents commitment to a new future. In contrast, prolonged analysis of existing or historical practices (implicit in many of the articles in this issue), even though undertaken with the goal of future change, inevitably feels rearward facing
Coughlan et al. [66, p. 132]

As already mentioned before, this aligns with the goal of thesis overall – to branch out into the future, to create new ways of how to onboard software developers. In addition to that there is a need to research a multitude of applied implications. Overall, this points at a designerly approach being the right methodological choice here. Critics might argue that such an approach is missing rigor and reproducibility. While especially the aspect of reproducibility cannot be denied, it has to be said, that ‘Design research conducted according to strict scientific procedures can produce highly valuable knowledge’ [64, p. 60]. To actually create this valuable knowledge, a strict scientific procedure has to be chosen and applied. This is where RtD comes into play. The term was first coined by Sir Christopher Frayling [67] and brought into the field of Human-Computer Interaction (HCI) by Zimmerman et al. [68]. They describe RtD in HCI as a way to produce novel integrations of research into HCI to create something, that represents the preferred state of the world [68, p. 493]. RtD as a whole can be described more broadly as a way to conduct scholarly research with methods originating from design practice [69] ‘as a legitimate method of inquiry’ [70, p. 310]. Zimmerman et al. also reinforces the arguments on why RtD would be an adequate methodology in the case of this thesis:

RtD allows researchers to rely on designerly activities as a way of approaching messy situations with unclear or even conflicting agendas; situations that are not well suited to other methods of inquiry. Additionally,

RtD forces researchers to focus on research of the future, instead of on the present or the past.

Zimmerman et al. [70, p. 310]

Zimmerman also proposes criteria for evaluating RtD approaches, to ensure that as much scientific rigor as possible is applied to such approaches. The four criteria proposed are:

- **Process** – ‘In documenting their contributions, interaction design researchers must provide enough detail that the process they employed can be reproduced. In addition, they must provide a rationale for their selection of the specific methods they employed’ [68, p. 499]
- **Invention** – ‘The [...] contribution must constitute a significant invention [...] researchers must demonstrate that they have produced a novel integration of various subject matters to address a specific situation’ [68, p. 499]
- **Relevance** – ‘researchers must also articulate the preferred state their design attempts to achieve and provide support for why the community should consider this state to be preferred’ [68, p. 499-500]
- **Extensibility** – ‘defined as the ability to build on the resulting outcomes of the interaction design research: either employing the process in a future design problem, or understanding and leveraging the knowledge created by the resulting artifacts’ [68, p. 500]

Delivering on all those aspects is crucial in order to allow for valuable knowledge generation despite not following a traditional research approach. Therefore, these criteria are going to be referred back to throughout both the study and the results of this thesis, starting with the process. Creating such a process from scratch could weaken the study’s results, as an unproven approach has to be validated in itself (independently from the results of its application). Thus, following and documenting a proven process as part of a proven set of methods is necessary here. Within RtD a multitude of different methods emerged since its integration into HCI. Fortunately, Sanders condensed years of research on design research as a whole and condensed it into a map of said design research, as visible in *Figure 3*.

It is important to note here, that this does not show a map of RtD methods specifically, but a more complete one of many methods applied in HCI. The two axis of the map

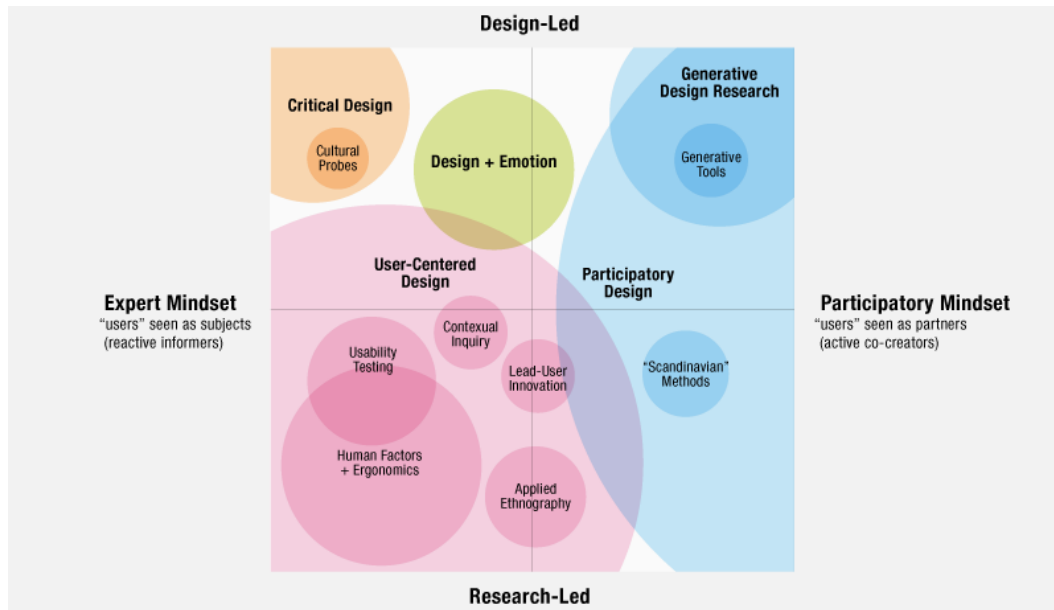


Figure 3: ‘Map of design research – research types’ by Liz Sanders [71, p. 3]

describe firstly, which research type the methods align to – either design-led or research-led, based on the distinction described earlier. Secondly the so-called *mindset* focuses on if an expert creates something that then is relayed to the users or if the users are active co-creators as Sanders puts it [71, p. 5-6]. RtD can be located mainly in the upper-left quadrant of this map, although it can be argued that some methods include the user as a more active part. Generally, though, RtD is based upon careful preparation by experts and therefore gravitates towards the aforementioned expert mindset. In *Figure 4* Sanders extended her original map with some more methods.

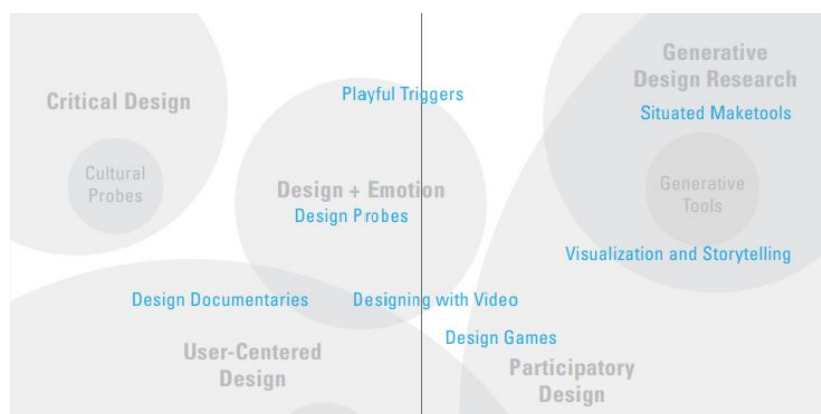


Figure 4: Upper section: ‘Map of design research – new tools and methods’ by Liz Sanders [71, p. 6]

To narrow down the methodological choices it is crucial to estimate where this study would fall within the presented map. Concerning the research-led vs. design-led, the

case has already been made for a distinctly design-led approach. Regarding the mindset the answer is a bit less obvious. While we do want to gather information directly from developers in actual onboarding settings, this does not necessarily mean that they are participating in the design process itself. Most of the work on the theoretical foundation design is already done, therefore at that step, there is need for additional input. Where additional input *is* needed, is at the transition from implemented implications towards recommendations and starting points for future research. An approach where both the expert mindset and partly active participation is possible, would thus be the desired one. On the map methods allowing for this would subsequently be located in the middle of the upper section.

In that section we can find the two methods *Playful Triggers* and *Design Probes*. The term *playful* strongly suggests that this might be the right method to research play. Upon closer inspection it might not be fully applicable to this study, as ‘This methodology utilises playful, tactile, everyday qualities of objects’ [72, p. 16] and then ‘take on the meanings placed on them by the participants’ [72, p. 16] to facilitate conversation. This would align with the goal of researching how developers experience (and how open they are towards) such playful triggers in onboarding processes, but disregards the implementation part of the study. In addition to that tangible interaction together with facilitated on-site collaboration and conversation due to ongoing work-from-home mandates especially for software developers due to the COVID-19 pandemic. Therefore, it is required that the chosen method can be executed remotely. A closer look into *Design Probes* consequently is the next step. As these exist in many variations, these have to be discussed in order to choose a suitable of these variations.

3.3 Probing in a Digital Context

Before delving into these variations, the origin of probing has to be mentioned, as it is going to help in understanding the process and goals of what came after. This origin can easily be traced back to Gaver et al., who first mentioned *Cultural Probes* as a research method in 1999 [73]. Gaver et al. were tasked with researching novel interaction techniques for the elderly within their local communities. The difficulty they were faced with, was the cultural difference between them as the designers and

experts and the elderly from different parts of Europe. To overcome these difficulties, they developed these so-called cultural probes, ‘packages of maps, postcards, and other materials [...] designed to provoke inspirational responses from elderly people in diverse communities’ [73, p. 22]. According to Gaver et al. these probes ‘address a common dilemma in developing projects for unfamiliar groups’ [73, p. 22]. The aforementioned parts of the probes (postcards, maps and more) were designed to be approached casually and to spark open-ended discussion. The probes consisted of two other parts as well. Firstly, questions were included, that concerned the environment of the elderly. These were as open-ended as possible in order to give the participants as much room as possible to answer casually. Secondly the probes included a self-documenting aspect. Participants were tasked to take record specific actions daily for a week [73, p. 22-26]. Concerning the particular goals, Gaver et al. also mention gaining ‘new understandings of technology’ [73, p. 25] and designing to ‘shift current perceptions of technology functionally, aesthetically, culturally, and even politically’ [73, p. 25]. Another important aspect is that the resulting data is not guided by any single objective problem statement, instead it is inspirational data that comes out of the process. The actual launch of the probes was then done in meetings at the local sites, after which the participants used the probes to self-document as described earlier. Finally, Gaver et al. also describe how, even though ‘the probes were central to our understanding of the sites, they didn’t directly lead to our designs’ [73, p. 29]. Cultural probes are widely used nowadays in HCI research [74, 75, 76, 77, 78]. So widely used in fact that multiple variations developed from these cultural probes.

One that is particularly interesting in the context of researching digital artifacts are *technology probes*. This term was coined by Hutchinson et al., who originally used it as a way to design technologies for and with families [79]. They deployed two different technology probes in two different contexts, a *VideoProbe* and a *MessageProbe*.

Independent from these different probes, what Hutchinson et al. wanted to achieve was to get knowledge and user feedback early on in the design process. Not only that, they wanted ‘them to be active partners in the design process’ [79, p. 2]. Although it has to be noted, that Hutchinson et al. did not go so far as to ‘expect the family members to become designers’ [79, p. 18]. It is also mentioned where exactly these *technology probes* differ from *cultural probes*, ‘cultural probes tend to involve a single activity at a

particular time and do not stress technology per se' [79, p. 18]. The ultimate goal of a *technology probe* is similar to that of traditional *cultural probes*, albeit more focused on a specific technology: 'Ideally, technology probes will spark new ideas and help the families articulate ideas for the prototypes we will build' [79, p. 18]. To create a *technology probe* that can achieve such a goal, Hutchinson et al. define the five following features:

- **Functionality** – 'Technology probes should be as simple as possible, usually with a single main purpose and two or three easily accessible functions' [79, p. 18]
- **Usability** – 'Technology probes are not primarily about usability in the HCI sense, so during the use period, we do not change functions' [79, p. 18]
- **Logging** – 'Technology probes collect data about relationships within the family and help family members (and us) generate ideas for new technology' [79, p. 18]
- **Flexibility** – 'technology probes [...] should be designed to be open-ended with respect to use' [79, p. 19]
- **Design Phase** – 'Technology probes are intended to be introduced early in the design process as a tool for challenging pre-existing ideas and influencing future design' [79, p. 19]

These would be the criteria a possible technological implementation in the upcoming study strives for. This is in contrast to a prototype, that is much more feature rich from the beginning, focuses on usability and improved iteratively (rather than implementing something new based on the data gathered through a probe) [79, p. 18-19]. Generally speaking one could describe these technology probes as a kind of middle ground between, cultural probes with no preceding implementation, and the evaluation of full-blown prototypes. This is also in contrast to so-called *digital cultural probes*, as e.g., Iversen & Nielsen applied it [80]. They went for more of a traditional approach with a stark focus on self-documentation and content generated by the participants, rather than deploying an implementation for them to use.

At this point, the questions arises on what to take from all of that regarding the methodological decision at hand. Answering that question requires contrasting the goals of our research with what probes in general and *technology probes* in particular provide. The overall goal of the thesis is to explore new opportunities for playful approaches in an onboarding context as mentioned numerous times before. Thus, there

is a need for a method that at its core is exploratory. Probes certainly can provide that, as Mattelmäki describes in her doctoral thesis on *Design Probes* as whole. She writes, that ‘probes have an exploratory character. They explore new opportunities rather than solve problems that are known already’ [81, p. 40]. Another important point mentioned mainly in RQ5 is the need to research how software developers specifically experience playful aspects. Researching this could arguably be the biggest strength of probing. Mattelmäki describes it as follows, ‘probes look at the user’s personal context and perceptions. The purpose is to outline human phenomena and users, as well as introduce the user’s perspective to enrich design’ [81, p. 40]. This is also described by Gaver et al. as ‘The real strength of the method was that we had designed and produced the materials specifically for this project, for those people, and for their environments’ [73, p. 29]. *Technology probes* in particular can provide even more value in the case of this thesis. RQ4 and RQ5 also call for testing out implementations of the design implications in real-world settings, while still being open towards new input from developers. Referring back to Hutchinson et al., they describe the goals of *technology probes* as:

the social science goal of collecting data about the use of the technology in a real-world setting, the engineering goal of field-testing the technology, and the design goal of inspiring users and designers to think about new technologies

Hutchinson et al. [79, p. 17]

This, again, closely aligns with the research goals of this thesis – there is the need for collecting data about the design implications in *actual onboarding settings*. There is also the need to *field-test* these digitally implemented implications. Finally, there is also – as mentioned before – the need to still be open about new input, enriching what was discovered in literature. Overall, the clearly visible alignment of goals point towards *technology probes* as a method of choice for the study. The relatively good adoption of this method (e.g. in [82, 83, 84]) further manifests it as an adequate choice. Now, the only thing left is to define a rough overall process for the execution of this method. That process is then elaborated on in great detail in chapter 4 to ensure the study manages to meet the quality criteria for a well done RtD approach, as described by Zimmerman [68, p. 499]. It is crucial to mention some environmental constraints to

the study here. Firstly, we are working with digital source code with which the participants (software developers) have to (mostly) interact with in a digital environment. Secondly, as acknowledged before, as of June 2021, there are many software developers choosing or being ordered to work from home due to the COVID-19 pandemic. Therefore, it is necessary that the chosen process can be done fully remotely. There is no single process to follow when designing, executing and analyzing *probes*. Gaver et al. even mentioned, that the ‘probes were not designed to be analyzed, nor did we summarize what they revealed about the sites as an explicit stage in the process’ [73, p. 27]. Hutchinson et al. on the other hand mentioned *logging* and analyzing these logs as a distinct part of the process, as described before. How this is done specifically differs from study to study. Edwards et al. for example, logged user-generated data and created accompanying questionnaires to collect quantitative data on the participants use of their probes [84, p. 107-110]. Mattelmäki proposes personal interviews to ‘extend, supplement and revise the signals collected in the probes’ [81, p. 86] where qualitative data is generated. Ultimately, there is great variance in the execution as well as the analysis of these probes, as such it is even more important to carefully lay out every part of the final process.

Overall, there are three distinct areas that are part of the process, the development & implementation of the probes, the execution & deployment of the probes and the analysis of the data generated through the probes. Concerning the development of the probes, the path is pretty clear, as the features within the probe should be based on the identified design implications. The specific process of how these were transformed into probe features is therefore described in great detail in chapter 4.2 and specifically in the sub-chapter 4.2.1. After the *technology probe* with a set of features is developed it then has to be deployed into the context and executed by the participants. Therein it is important to gather the data needed to answer the research questions. Generation of data on personal experiences, as well as exploratory data is paramount. This strongly suggests collecting qualitative data, as it is uniquely able to gather perspectives of participants [85, p. 7-8] as well as generating new knowledge on a setting too pluralized for quantitative approaches [85, p. 4-5]. That does not mean there is no structured data at all, though. As *logging* is such an integral part of *technology probes*, user interaction with the features of the probe is going to be logged accordingly. To get

an even more complete understanding of how the individual participants experienced the probe and to gather qualitative input on new ideas from participants, qualitative questions are going to be asked after probe execution. Additionally, to get a sense of how effective the probe features were in actually transmitting information about a project, questions on the project at hand are asked as well. This allows for a rich set of data for each participant to analyze. The focus strongly lies on generating as much valuable data as possible for each participant rather than aiming for a maximum number of participants. Overall there is going to be a three-step process concerning the probes:

1. Probe Development
2. Probe Execution & Data Collection
3. Probe Analysis

As mentioned, the specific tasks within these steps are going to be described in great detail with every additional process choice documented as well in chapter 4.

Concerning the probe analysis, the methods of analyzing the generated data is subject of the following chapter.

3.4 Data Analysis

The last high-level methodological choice to make is concerned with how to analyze the data gathered by the probing. To recapitulate, data is going to be gathered through these four methods:

- **Demographic Information** – Prior to the probes some information on the professional background of the participants is gathered to gain a comprehensive picture of the participants
- **Participant Interactions** – Throughout the probe the actions taken by the participants are going to be logged and categorized in an automated way (e.g. a click on an element on the screen)
- **Questions on the underlying project** – The participants are going to be asked questions on the project they are onboarded on to
- **Questions on the research topic** – The participants are going to be asked questions centered around the research topic to gather their expertise and ideas

on the topic

In order to correctly analyze that data, it is crucial to know what kind of data is generated by each of these. Only after that a suitable method of analysis can be found. Starting with the demographic information, which can easily be identified as fixed data types (e.g., age always equals a number). These do not have to be analyzed on their own, they rather set the data in an individual context for each of the participants. The interactions on the other hand are going to be automatically logged during the study. Therefore, the logs need to follow a predefined structure in order to log a variety of different interactions. The actual structure of these interactions is described in chapter 4.3 as it also depends on the technology used to capture these events. The interactions could be analyzed in a way to find patterns of interactions, like analyzing the number of clicks at a certain point in time. This might be very useful to delve into specific aspects of the probe and analyze them in great detail (see Fitton et al. for example [82]). While this can be useful for intricate features, these are not really the part of the probes, as it should be kept simple. In addition to that evaluating the effectiveness and details of these particular features is not the prime goal of this study. Nonetheless, it can help in solidifying arguments based on the other kinds of data and can be referred to together with the participants feedback. Because of that the interactions together with the time of that interaction is going to be logged but only analyzed in detail as needed depending on the other data. Simple aggregations of total interactions and total time spent within the probe are generated from the raw logs, though, to give a sense of how active the in-probe interaction was. Moving on to the questions on the underlying project, these as well should help in understanding how the participants experienced the probe. The goal of these questions should be to understand if some of the underlying information was transported successfully. The questions should be crafted in a way, that allows to verify if the participants recognized the project information. Lastly, the arguably most important part is to gather data on the experiences of the participants, their attitude towards playfulness in onboarding processes and their ideas on how to integrate it. As already described in the previous chapter a qualitative approach was chosen for that. Concerning the actual creation of the questions, these are centered around the specific topic at hand, similar to how problem-centered interviews are structured around an issue [86]. The actual data

collection for these problem-centered interviews revolve mainly around in-person enquiries, mirroring and lively discussion [85, p. 232-236]. Due to the study being a remote one, such a lively discussion might be at harder to execute. Therefore online interviewing, as mentioned by Flick [85, p. 243] could be a more adequate solution. This type of interviewing can be done in a few different ways. Probably, the two most important decisions concerning the study design are, if the questions should be answered asynchronously or synchronously, and through which medium the questions are asked. This largely depends on the study setup, as it could be possible to integrate the questions within the probe or the medium over which the probe is interacted with. Thus, the final decision rationale is described in the subsequent chapter. This final decision does not change the method to analyze the data, as this data is unstructured text that is going to be used for qualitative argumentation. Depending on the amount of data gathered there are different options for the following analysis. From using the participant directly to support argumentation to applying a specific method of analysis, there is a broad range of possibilities. On the more rigorous end of the scale, two of the most notable approaches are thematical analysis [87] and content analysis [88]. These mainly differ in their starting point, where themes are generated from text (thematical analysis) or a text is analyzed with a set of categories in mind (content analysis) [85, p. 490]. Concerning the research done within this thesis, a lot of themes and implications were already discovered, therefore it might make a lot of sense to use them as a starting point for the analysis. This should make content analysis a suitable choice for analyzing the data. Executing such a content analysis procedure involves multiple steps, as described by Schreier and Flick ([89, p. 174], [85, p. 483-485]):

1. Selection of the material
2. Creation of a category system
3. Segmentation of the material
4. Main coding
5. Interpretation and Presentation

Within those steps, additional methodological decisions and considerations have to be made. Concerning the first step, the respective selection is an easy one to make. As the study is specifically crafted in order to generate qualitative data, all the qualitative answers within that study are subject to the analysis. Thus, all answers by each

participant on the research topic and on the underlying project are selected for the content analysis. The additional data collected throughout the probing is used as a support for the result argumentation, as was mentioned before.

Regarding step two, the creation of a category scheme, there are three distinct options of coming up with these categories: *Deductive*, *Inductive* and *Deductive-Inductive*. The decision on one of these approaches mainly depend on if the result argumentation should be derived from the material (inductive) or defined by literature and brought to the material (deductive). Mixed approaches exist as well, where some categories are brought to the material and some are based on the material (deductive-inductive), by defining top-level categories from literature and lower-level categories from literature for example [89].

This is what is going to be done in the case of this study, with the overall direction of the analysis defined through the literature and the research questions, while also allowing for new categories and themes based on text.

Concerning step three, a unit of segmentation has to be defined. Two options – that can again be mixed and inform each other – are coding units and contextual units.

Coding units carry meaning by themselves and can be definitively allocated to a category, while contextual units only help in giving meaning to the coding units [89].

In the context of this study, the material is divided into coding units. Where necessary or helpful to give context, contextual units are used. These contextual units can either be the features/questions they refer to, but also the logged data, where it supports the argumentation.

The coding itself is then done as a two-step process, with marking important units at first and then coding these marked units in a second step, as described by Schreier [89]. The interpretation and presentation ultimately can be found in detail in chapter 5.2 onwards.

4 Research Design

After laying out the methodology used throughout the study, the actual application of the methodology has to be laid out. This is going to be the subject of this chapter. In the following considerations on research ethics, as well as participant selection are described. In addition to that, the development of the technology probes, the process for going through these probes, the technological setup for data collection, and the data analysis approach are documented in detail. The study approach as a whole is then summarized at the end of this chapter.

4.1 Research Ethics

Before delving into the specifics of the study an important, sometimes neglected, aspect of the study design has to be talked about: *Research Ethics*. In order to do an ethically sound study with participants some important principles have to be met. Flick proposed eight principles of ethical research, that this study aims to follow (taken from Flick [85, p. 135]):

1. *Researchers have to be able to justify why research about their issue is necessary at all*
2. *Researchers must be able to explain what the aim of their research is and under which circumstances subjects participate in it*
3. *Researchers must be able to explicate the methodological procedures in their projects*
4. *Researchers must be able to estimate whether their research acts will have ethically relevant positive and negative consequences for the participants*
5. *The researchers must assess the possible violations and damages arising from doing their project*
6. *The researchers have to take steps to prevent violations and damages identified according to principle 5 above*
7. *The researchers must not make false statements about the usefulness of their research*
8. *The researchers have to respect the current regulations of data protection*

To meet these requirements towards ethics participants have to be informed on each of the principles mentioned above. To achieve this, both a form of consent and a data protection information sheet have been created, that can be found in appendix A. Principles 1 through 3 should be met, by describing in the beginning paragraph and the subsections *General Study Information* and *Overall Study Procedure* of this form. Concerning principles 4 through 6,

there should not be any positive or negative consequences for the participants, as the study only consists out of a virtual probe with questions only on the expertise of the participants, not on personal or intimate details. Should the participants wish to participate anonymously, an option is given as well. If the participants check this option in the online form, the gathered data is stored by using an random identifier number not connected to their name. As for principle 7 no false statements on the goals of the research are made, the goals match those mentioned within this thesis. Finally, the last principle is met by including all relevant information on the specifics of data processing in the subsection *Confidentiality and Data Processing*. Therein everything that is logged is described together with where and how it is logged and processed. To make it clear for participants, that they can withdraw their participation in the study whenever they want, an additional section (*Rights*) is added to the form. The resulting form is then presented to every participant before they start taking part in the probe and without consenting to it the probe cannot be started.

In addition to the considerations that are part of the consent, due to the current, ongoing COVID-19 pandemic there are additional considerations to be made. These mainly revolve around not requiring the participants to take part in a study on-site. As mentioned before, the probe is going to be done remotely, therefore, there is no need for additional precautions have to be taken on that account.

4.2 From Theory to Probe

The most important part of the research design arguably is the actual technology probe. As described in the methodology section, this requires careful attention to the features described by Hutchinson et al: *Functionality, Usability, Logging, Flexibility, Design Phase* [79, p. 499-500]. The functionality is determined by the design implications found in literature and described in detail in the following chapter. Concerning the usability, Hutchinson et al. only state that there should not be any incremental changes during a probe, therefore this is not further elaborated on. Logging is going to be part of the data collection phase of the probe and therefore is described in the respective chapter. Finally, concerning the design phase, the probe is developed as the very start of the design process within this thesis but also as the start of further implications in this line of research.

A crucial part of developing the technology probe is the transition from theory to a finished probe to the execution of that probe. Therefore, this chapter is mainly concerned with the tasks done and decisions made in between those steps. At the start of this process is the transformation from the design implications and attributes of play discovered during

literature research into actionable probe features. This again is divided into multiple sub-steps, the first of those steps being an ideation phase. Within that ideation phase, the design implications are used as a starting point based on which possible features, playful elements and game mechanics are ideated upon. This is loosely based upon IDEO's brainstorming principles [90], as can be seen in *Figure 5*.

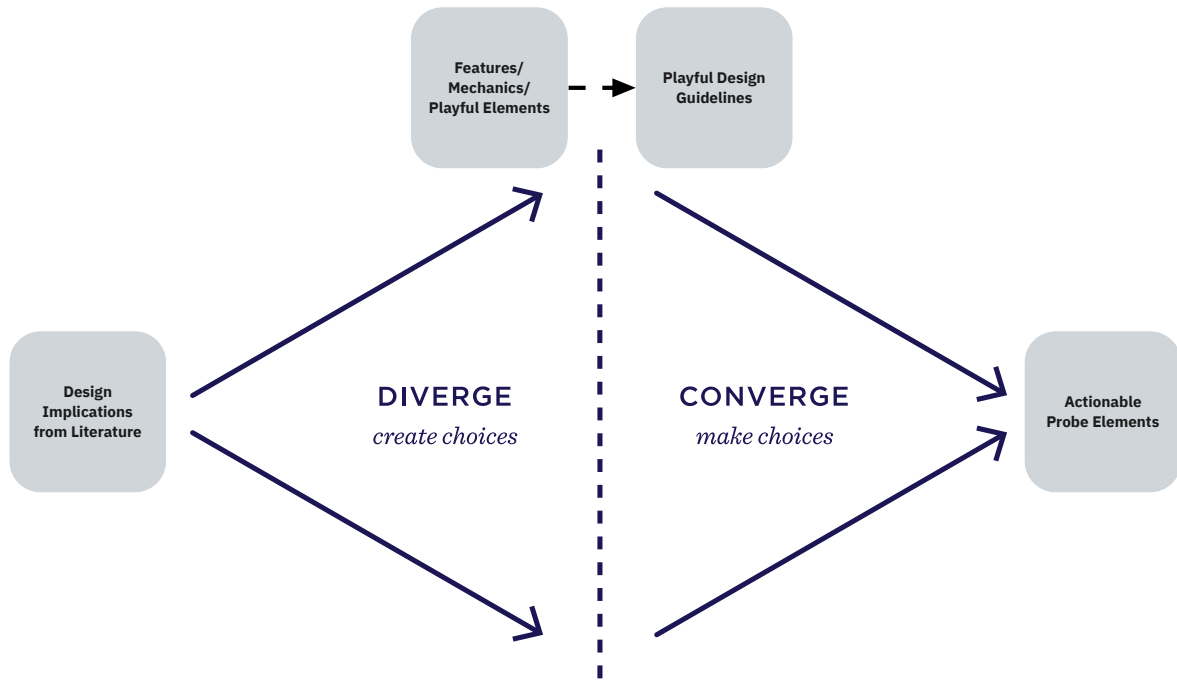


Figure 5: Adapted IDEO Brainstorming Process (originally by IDEO U [90])

Starting with the design implications as much features/concepts/playful elements as possible are ideated upon to generate a variety of technological choices as possible rooted in literature. With those generated, the next step then is to *make choices*, meaning, deciding on a few that then can be implemented within the technology probe. Concerning this thesis, the respective brainstorming process is visualized and documented in *Appendix B*. Therein you can see the implications on the far left of the diagram, based on these a braindumping session [91] was held. The result of this session is visualized to the right of the attributes of play within *Appendix B*. Examples of the ideas gathered are:

- Serendipitous random effects within the game/probe
- Generate game objects for contributions and possible tasks
- A Stage where the players can explore for themselves

For the full list, refer to the respective diagram in the appendix. What is also shown in this diagram, is on what kind of design implication the respective idea is based upon. These ideas

were then condensed into actual guidelines that can inform the probe implementation or any kind of playful design for that matter. After that converging these guidelines is necessary. This is done for two reasons. One, to reduce the number of features in order to conform to the technology probe attribute of *functionality* described by Hutchinson et al. [79, p. 18]. Two, in order to create actionable tasks rather from the very vague ideas and slightly less vague guidelines. The results of this process can be seen in detail in *Appendix B*. The converging itself was done by combining and reducing the guidelines so that an implementable task description is reached. If some guidelines were not included as features, these were dismissed by me based on the perceived difficulty of implementation. That is because the technology probe should be kept simple and focused on gathering as much valuable data as possible rather than being an intricate and detailed prototype implementation.

Another important decision to be made at this point is which onboarding level the probe should focus on. Should it be on the source code/programmatic level, should it be at the process level or should it be at a social & developer-centered level. This decision depends on a few things. At first, it has to work in a remote, decentralized settings due to the constraints defined in the previous chapter. Secondly, it has to be “implementable” by me, based on the timeframe and my previous experience. Thirdly, and arguably most importantly, the probe should be able to collect data, that contributes and extends this line of research. Concerning this third statement, there is no actual constraint imposed by existing research, as on none of the described onboarding levels, there are existing implementations in research. The first and second statement on the other hand do impose some constraints. Because of the remote setting, a social setting might be hard to replicate remotely. In addition to that, as is going to be described in chapter 4.5, the study is not going to be done within one single team of developers working on one project, rather the probe is targeted towards individual developers. This further increases difficulty in deploying a probe within a social setting. Therefore the probe is rather going to focus on program comprehension and process-centered techniques like information seeking and feature location, which can be implemented on a source code/project level. Nonetheless, research into the other levels of onboarding could be equally valuable in the future.

At this point most of the non-technical decisions have been made. From the design implications, an ideation phase, followed by a phase of converging, produced a set of possible features that can be implemented. As mentioned before, these are shown in the second-to-last column in *Appendix B*. Actually deciding on the feature set used in the probe, depends on some technical decisions, though. This is why in the following chapter there is going to be a

deep-dive into all technological considerations including the final feature set.

4.3 Probe Implementation

To actually implement a coherent technology probe, these considerations have to be discussed in detail, starting with the choice of technology in which to implement the probe. As described in the previous chapter, the probe is going to target the psychology- and process-centered level of onboarding and is based on the source code and meta-information of a project. Best-case this probe could be applied to any project independently from its technology. Depending on the programming language and framework there are grave differences in project structure, though. A *Ruby on Rails* ⁹ project for example, gives the developer a fixed structure out of the box, while there is no such structure when developing a *node.js* ¹⁰, project with *express.js* ¹¹. Creating a probe that would work with both of those examples, or even more than that, would not be feasible in the given timeframe. More importantly, for a probe meant to be executed once by each participant, there is no benefit in investing additional implementation time into allowing the probe to work on multiple, differently structured projects in succession. More specifically, in the case of this probe the underlying project is not going to be changed for each participant to achieve some amount of comparability between the experiences of the developers. Deciding on which specific project to choose depends on some constraints as well, though.

First of all in order to build a probe, that can be sent out to developers not necessarily working within the same company and/or team, the source code has to be publicly accessible. This rules out projects proprietary to a single company or team right away. In addition to that the project should cater to the knowledge of a wide variety of developers. Therefore, the technology used within the project underlying the probe should be popular enough, that a general understanding of the project setup is possible for the participants. According to a survey done by StackOverflow Insights ¹² with 83,052 responses on the topic of programming languages developers have done extensive development in the last year, there is a clear leader in popularity (multiple responses by developers were possible): *Javascript* (64.96% agreement), its server-side framework *node.js* (33.91% agreement) and its superset *Typescript* (30.19% agreement), with the closes other technology/programming language being *Python* (48.24% agreement) [92]. Thus, choosing a *Javascript*, *node.js* or *Typescript* project, might be

⁹<https://rubyonrails.org/>, accessed on 10th of August 2021

¹⁰<https://nodejs.org/en/>, accessed on 11th of August 2021

¹¹<https://expressjs.com/>, accessed on 11th of August 2021

¹²<https://insights.stackoverflow.com/>, accessed on 11th of August 2021

the most approachable for a wide variety of developers. The only downside to choosing such a project, is that none of these technologies by themselves give a fixed structure to rely on when building a probe upon it. Something that gives that structure, is the concept of monorepos. These are repositories, where the ‘codebase is contained in a single repo encompassing multiple projects’ [93, p. 226]. This gives inherent structure to the project, with different subprojects as a whole to explore on different levels. Also, larger codebases might profit more from onboarding help due to the increased complexity going hand in hand with the increase in size. Moreover, in a well-crafted monorepo, any ‘project in the repo can be built only from dependencies also checked into the repo’ [93, p. 226]. This allows for visualizations of these connections within the monorepo and how subprojects relate to each other. Especially so, when combined with *Javascript*-based monorepo-organization tools like *Lerna* ¹³. *Lerna* stores information on the dependencies within the monorepo in two files, `package.json` and `lerna.json`. In addition to that the subprojects are guaranteed to be found in a `/packages` directory. This further allows for reading these files and directories in code in order to easily implement probe features on top. Consequently, a Javascript/Typescript-based monorepo that uses *Lerna* or a similar tool, with predefined structure, where also the source code is publicly available has to be found. Fortunately, the largest provider of publicly available repositories, *Github* ¹⁴ makes it possible to filter for these criteria (filter results can be found here ¹⁵). A highly popular monorepo, that is well maintained and actively collaborated on, is the *ethereumjs-monorepo*, a repository containing the *Javascript* implementation of the *Ethereum* blockchain ¹⁶. While this repository does not use *Lerna*, it still follows the structure imposed by *Lerna*, making it possible to programmatically read the monorepo information from its directories and `package.json` file(s). Because of that and the repository matching the other constraints imposed on this decision, the *ethereumjs* monorepo is chosen as the underlying project for the implementation of the technology probe.

The next decision to make is then the one of with which technology and in which in environment to create the technology probe. Due to the setting as a remote probe, the technology used for the probe should be easy to distribute to the participants, without any additional setup work on the researcher’s, but also on the participant’s side. The perhaps easiest way to achieve this, is to create an interactive web page, accessible through just an browser without any additional installation steps required. The downside of this approach

¹³<https://lerna.js.org/>, accessed on 11th of August

¹⁴<https://github.com/>, accessed on 12th of August 2021

¹⁵<https://github.com/topics/monorepo?l=typescript>, accessed on 12th of August 2021

¹⁶<https://ethereum.org/en/>, accessed on 12th of August

would be, that the probe is removed from how most developers interact with the source code of a project – through an IDE, text editor or the command line. Due to multiple editors and IDEs existing, there would have to be a decision for a single tool. This in turn limits the number of developers being able to take part in such a probe or doing so in an environment different from what they are used to. In addition to that different editors and IDEs use different plugin systems, for which a non-negligible amount of previous experience or time invested is required in order to develop plugins. This, together with my extensive experience in a web environment and with web technologies led to the decision of implementing the probe in a web environment.

Based on these decisions, the final feature set can ultimately be defined. Referring back to Hutchinson et al.'s probe features, a further reduction of features is in order. Therefore, special attention is paid to decide on a set of playful features, that can be transformed into a coherent probe not overflowing with too much functionality and with definitive boundaries. What also went into consideration, was an attempt to match probe features with concepts from the underlying project, to foster it being used as an educational tool – similar to how Ian Bogost defined them [17]. This led to the implementation of the following features:

- **Visualizing the overall project structure** – Based on the idea that the code structure should be documented [54, p. 10] and that the micro and macro structure of projects should be visualized [45, p. 284], as well as Sicart's notion of visualizing creative appropriations of data [3, p. 28]
- **Free exploration within subprojects** – Based on the idea that playful objects should be freely interacted with (based on Sicart's notion in *Play Matters* [3, p. 40]) and the idea that players should explore and test without too much guidance (based on Sicart's notion in *Play Matters* [3, p. 57])
- **Implementation of a reveal mechanic to uncover objects in subprojects** – Based on the idea that users should complete intentionally ambiguous designs (based on Sicart's notion in *Play Matters* [3, p. 31]) and the idea that uncertain outcomes should be embraced, where the user can decide (based on Rodriguez as described in chapter 2.4 [14])
- **Discovery of potential issues** – Based on the idea, that specific tasks should be recommended to the newcomers in a project (based on [63], [42, p. 36], and [54, p. 8])
- **Discovery of important contributors** – Based on the idea, that social links in development projects are crucial and mentors should be recommended to newcomers (based on [54, p. 8] and [45, p. 284])

- **Discovery of 3rd-party packages used** – Also based on the idea of visualizing the micro and macro structure of a project [45, p. 284], where 3rd-party packages are part of both the code (=micro) and the overall structure (=macro) depending on the package
- **Creation of a companion helper** – Based on the idea, that a companion with personality enables playfulness (based on Sicart’s notion in *Play Matters* [3, p. 32])

These features can be combined into a coherent *game world*, that visualizes the project structure, but also enables the players to playfully explore specific parts of the project. While the companion feature deviates from this coherent set, it nonetheless can be implemented as a tool of communication and occasional helper for the other features. Overall this should allow the technology probe to be succinct enough in functionality (again referring to Hutchinson et al. [79, p. 18]) while also being open-ended (as mentioned by Hutchinson et al. as well [79, p. 19]) enough in paths towards the exploration.

With that decided upon, the actual implementation now can commence. This implementation can be split into two distinct parts. Firstly, the generation of the *game world*, and the discoverable elements of the probe from the underlying project. Secondly, the actual implementation of the user-facing, interactive part. These are described in detail as follows, starting with the user-facing implementation as the generation of the game world is based on how it should be visualized in the end.

4.3.1 The User-Facing Implementation

First of all a general idea of how the probe should be visualized aesthetically. Aesthetics play a vital role in the creation of any kind of probe. Gaver et al. point out, that they ‘believe aesthetics to be an integral part of functionality’ [73, p. 25]. Gaver et al. also mention, that a finish that is too professional could lead to the probe being perceived as too formal [73]. Thus, a balance between appealing aesthetics and a simple but functional visualization has to be found. To find this balance, several examples of a minimal but functional aesthetic were collected in order to create a rough direction forward in terms of visual design of the probe. These examples can be found in *Figure 6*. All of these have in common, that they are so-called *Minimalist Game Designs*, which in addition to some other characteristics, are defined through ‘abstract audiovisual representations’ [94, p. 38]. The characteristics of such games match the characteristics of technologies very well, as Nealen et al. point out, ‘these games feature a small set of mechanics and one core mechanic, while still being sufficiently deep and allowing for player exploration’ [94, p. 38]. This suggests, that going for such

minimal aesthetics might emphasize what the technology probe in itself is trying to achieve.

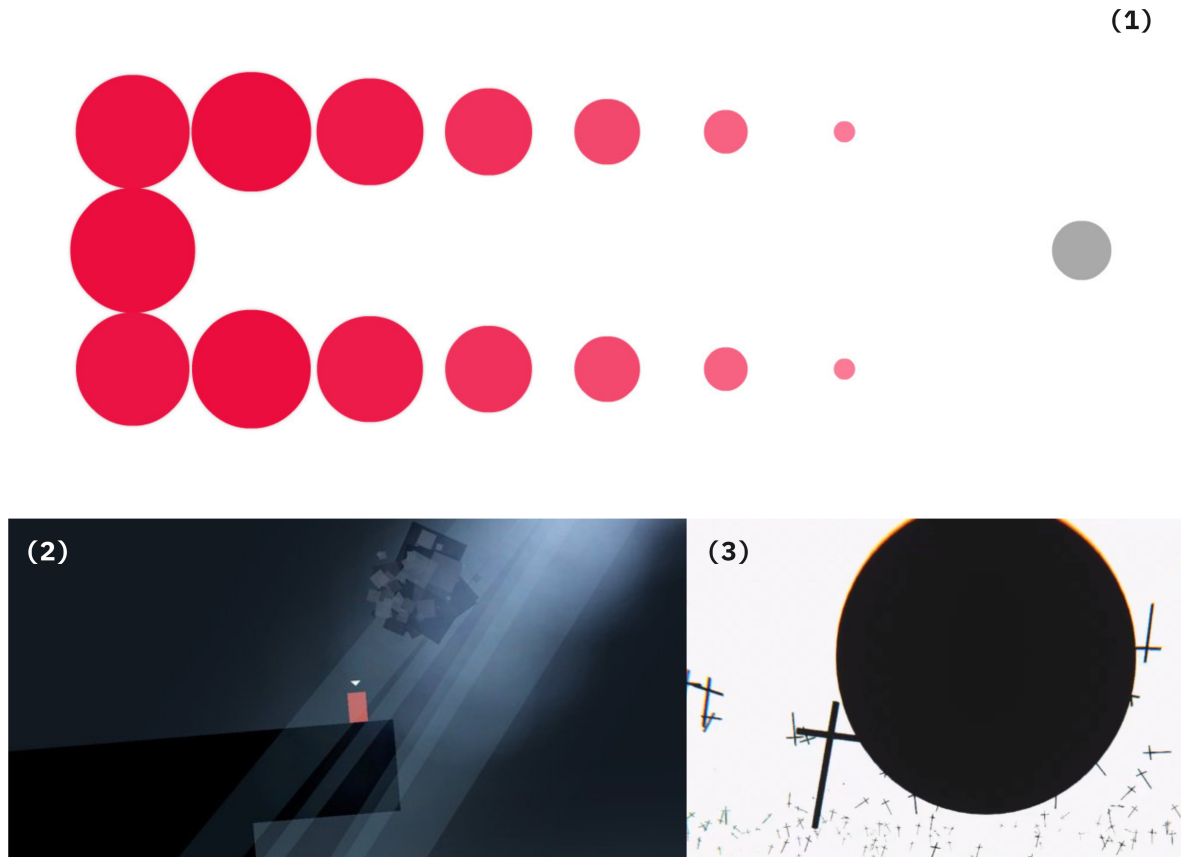


Figure 6: Probe aesthetics inspiration (including *Circles* [95] by Jeroen Wimmers (1), *Thomas Was Alone* [96] by Mike Bithell (2), and *Carthasis* [97] by Oleg Frolov (3))

With the general aesthetic direction decided on, the next step revolves around developing a world within which the probe features can be implemented. This game world consists of the game objects representing the different, discoverable objects, a character the player controls, and the background setting. The background setting is going to be kept as minimal as possible, in order to achieve the goal of a minimal, ambiguous but explorable setting for the player. The overall color scheme should strengthen these goals as well, therefore a monochrome color palette, with only shades of red as the accent color used throughout the probe. Concerning how the actual objects within the game world look, these are also visualized as simple as possible, similar to how *Circles* [95] is built. Rather than creating a visually intricate design, the playfulness comes from the interaction and from how the objects behave. All of the objects within the game world and their designs can be seen in *Figure 7*. With the main game objects, the color scheme, the overall aesthetic direction and the features out of the way, the actual implementation can be undertaken at this point. In order to do this, the technical foundation has to be decided on. As mentioned before, the probe is going

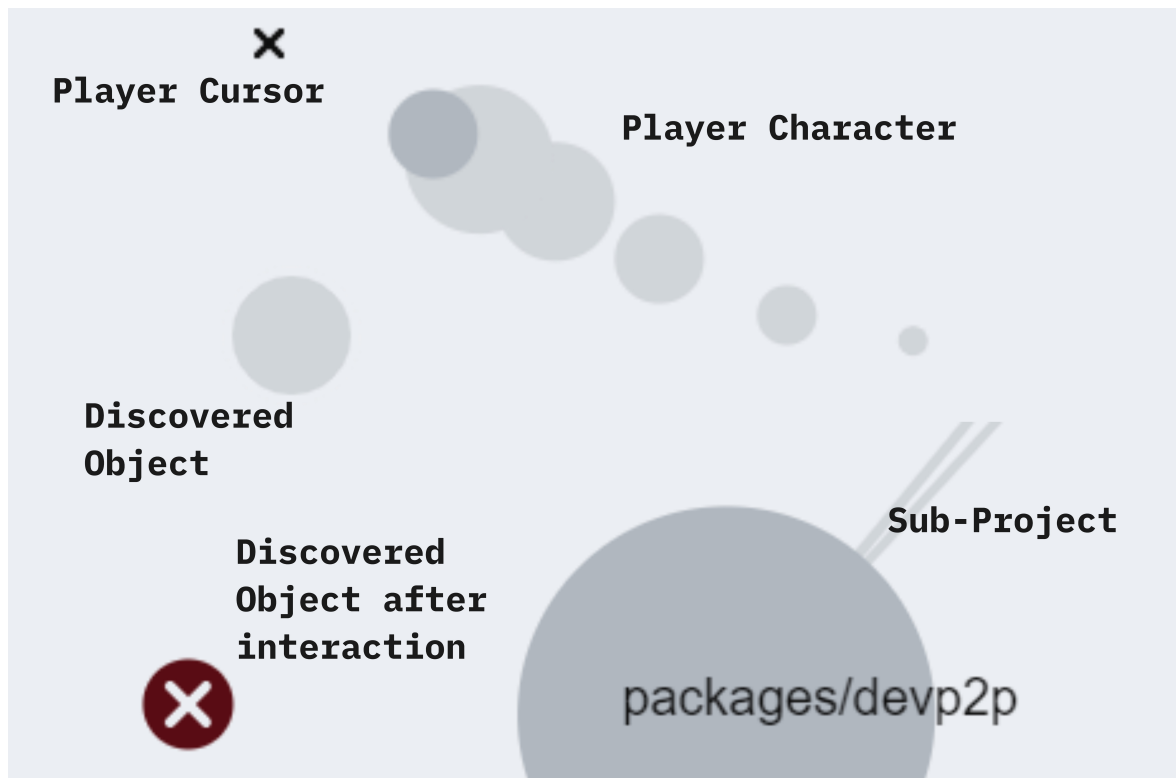


Figure 7: Game objects used throughout the technology probe

to be implemented as a web page, therefore the technology choice is limited to that space. Nonetheless, there is a huge variety of possible frameworks in that technology space to choose from. From entire game engines & frameworks to using no framework at all and relying on *Javascript*, *Hypertext Markup Language (HTML)*, and *Cascading Stylesheets (CSS)* alone. Using no framework at all – especially when implementing a highly graphical and interactive application – presents the need to re-implement a lot of features frameworks provide, like easy interaction handling, animation support and more. On the other hand a complete game engine brings much more, than is needed in this case. Due to the minimalistic visualization, there is no need for complicated physics replication or support for intricately designed sprites. A framework that sits in between those two extremes, is *p5.js*¹⁷, a *Javascript* implementation of the popular *Processing*¹⁸ framework. It excels in creating interactive visualizations, especially when based on geometric shapes. Combined with my previous experience using this framework, it presents a suitable choice for the implementation of the technology probe. In addition to that there are more libraries used for different parts of the implementation to speed up the development. As a build tool, that helps bundling and transforming the written source code into a browser-compatible package and allows usage of

¹⁷<https://p5js.org/>, accessed on 12th of August 2021

¹⁸<https://processing.org/>, accessed on 12th of August 2021

*Typescript*¹⁹, *Parcel* is used²⁰. To speed up development of all styling not connected to *p5.js*, *SASS*²¹ is used. In order to use a central data store, that houses and distributes the current game state, the libraries *Rx.js*²² and *zustand*²³ are used. Other important helper libraries, include *anime.js*²⁴ for animations outside of the interactive visualization, *Moment*²⁵ for date manipulation, *Lodash*²⁶ for general utility functions and *Firebase*²⁷ as a way to persist logging information and answers to the questions accompanying the probe. For the full list of all 3rd-party packages used throughout the probe, refer to the `package.json`²⁸ within the repository, where the probe's source code is managed.

All of these technologies serve the goal of creating the technology probe in the end. This creation itself can be divided into the already defined features, starting with the implementation of: **Visualizing the overall project structure**. This overall project structure is defined through the subprojects of the monorepo at hand. How the actual subprojects are programmatically read is subject of the next chapter. Important to note at this point, is the fact that this overview represents a distinct scene within the probe. Meaning, it is a separate view, where only the overall project structure is shown. To explore the contents of such a subproject, the game objects within this *overview scene* have to be interacted with, leading to a scene change to the subproject's *detail scene*. Within the *overview scene*, the game objects representing the subprojects are placed randomly onto the empty background (as can be seen in the respective file in the source code²⁹). To avoid too much game objects overlaying each other, the screen space is divided into sections according to the number of subprojects, within each of those sections one subproject game object is then placed. The size of these objects is determined by the size of the subproject, as is going to be described in the following chapter. The initial state of these game objects is shown in *Figure 7*. The player character (also shown in *Figure 7*), then is able to interact with the subproject objects. Upon collision of the player character with those objects, they grow in size and turn slightly opaque, letting the player still see their character, but also indicating, that an interaction with the respective object is possible. Another important visualization

¹⁹<https://www.typescriptlang.org/>, accessed on 13th of August 2021

²⁰<https://parceljs.org/>, accessed on 13th of August 2021

²¹<https://sass-lang.com/>, accessed on 13th of August 2021

²²<https://rxjs.dev/>, accessed on 13th of August 2021

²³<https://github.com/pmndrs/zustand>, accessed on 13th of August 2021

²⁴<https://animejs.com/>, accessed on 13th of August 2021

²⁵<https://momentjs.com/>, accessed on 13th of August 2021

²⁶<https://lodash.com/>, accessed on 13th of August 2021

²⁷<https://firebase.google.com/>, accessed on 13th of August 2021

²⁸See suppl. repository at path: `./technology-probe/package.json`

²⁹See suppl. repository at path: `./technology-probe/src/scenes/OverviewScene.ts`

shown between these objects themselves is the interconnection between subprojects of the same monorepo. These are indicated with lines connecting the subproject objects to each other. The actual implementation details would go too far into detail, but can be found in the respective file of the codebase ³⁰. Where the transition to the next feature of the technology probe happens, is after the interaction – a mouse click onto a subproject – with the respective object.

On this click, the *detail scene* containing the information for the respective subproject is shown. Therein it is possible to **freely explore the subproject** and to **use a reveal mechanic to uncover objects in subprojects**. Initially, the *detail scene* is fully void of any visible elements. Uncovering objects within the scene requires the player to use the reveal mechanic implemented within the scene. This reveal mechanic is loosely inspired by what games like *osu!* ³¹ are doing, where the player moves the character to different points on the screen, to reveal and interact with newly appearing circles growing and decreasing in size. In the case of this technology probe, the elements do not show up after a set amount of time, rather through user-induced reveal ‘*bubbles*’. These can be spawned at arbitrary points on the screen, growing in size until they fade out after a set amount of time. As soon as a *reveal bubble* is colliding with a currently invisible discoverable object, the discoverable object (as shown in *Figure 7*) can be interacted with, similar to the game objects representing the subprojects in the *overview scene*. To better visualize this interaction, *Figure 8* shows the player character, with a growing *reveal bubble*, revealing underlying game objects.

Within that figure, two different scenes are shown, with the first (1) one showing how reveal bubbles uncover the invisible game objects scattered on the scene. The second scene (2) in contrast, shows how – as long as, the object is within a reveal bubble – the discovered game objects signify, that they can be interacted with. In addition to this reveal mechanic, the discovered objects move randomly across the screen to turn a rather static uncovering experience into a more dynamic one – amplified by the fact that the discoverable objects only ever are visible during their collision with the reveal bubbles. Again the implementation details – like the collision calculation between *reveal bubbles* and the discoverable objects – can be found in the respective files of the accompanying probe codebase ^{32 33 34}.

On interaction with those discoverable objects, the nature of these objects is revealed. There are three different types of revealable information important to a newcomer, as discussed

³⁰See suppl. repository at path: `./technology-probe/src/sketch0bjects/Player.ts`

³¹<https://osu.py.sh/home>, accessed on 13th of August 2021

³²See suppl. repository at path: `./technology-probe/src/sketch0bjects/Player.ts`

³³See suppl. repository at path: `./technology-probe/src/scenes/DetailScene.ts`

³⁴See suppl. repository at path: `./technology-probe/src/sketch0bjects/Revealable.ts`

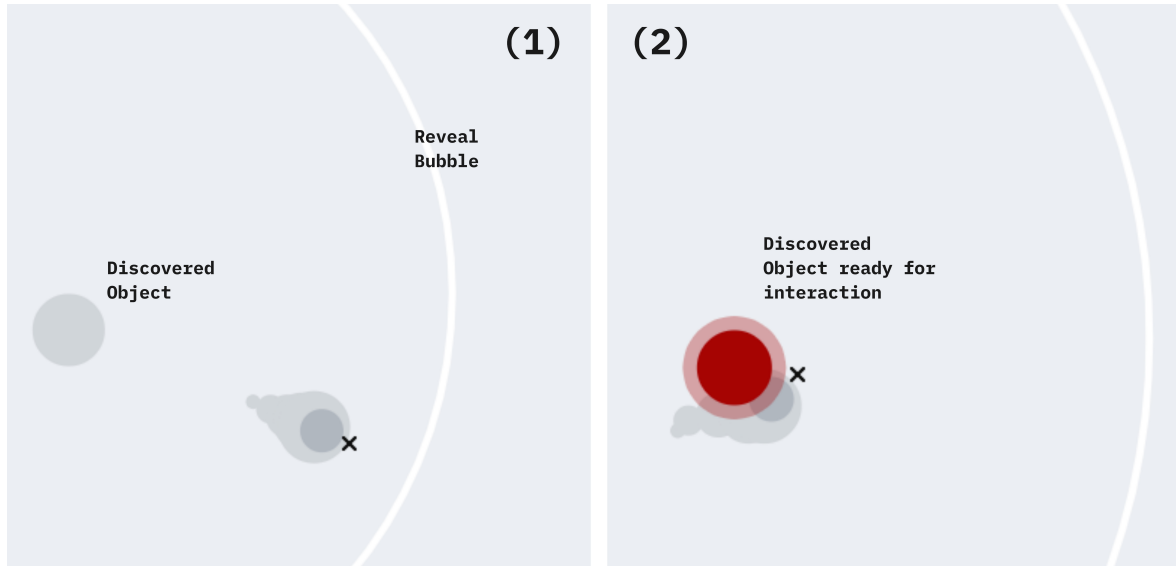


Figure 8: Probe detail scenes with discovered object (1) and object ready for interaction (2)

earlier: **The discovery of potential issues, important contributors & 3rd-party packages used.** Upon interaction with the respective discoverable object, an information message is shown. This message shows accompanying data, depending on the type of revealable information. The actual data that is shown is described in detail in the following chapter as it is directly generated from the underlying project. The visualization of the messages on the other hand is attached to this thesis in *Appendix D*. The player also gets the option to have a detailed look at the contributor's *Github* profile or at a potentially problematic file within the codebase. After the info message was dismissed by the player, the discoverable object is then permanently visible in state indicating, it has been interacted with (as shown in *Figure 7*). Again, the implementation details can be found in the respective source code file³⁵.

At this point, there is only one last feature to implement, the **creation of a companion helper**. To actually be helpful, this companion should guide the player through the intricacies of the probe and provide additional information when needed. Specifically, the companion indicates, when everything is revealed within a subproject and when everything is revealed within the probe as a whole. Additionally, if the player fails to interact with the elements on either of the scenes, the companion presents a message describing the game mechanic needed to accomplish the next step within the probe. To be precise, within the *detail scenes*, if after six seconds no reveal bubbles were spawned, the companion presents a message clarifying that mechanic. After ten additional seconds, if no discoverable object was interacted with, the companion tells the player to click the discovered objects. Within the

³⁵See suppl. repository at path: `./technology-probe/src/ui/info.ts`

overview scene, if after six seconds no subproject element was interacted with, the companion explains that mechanic as well. In *Figure 9* an example is shown of such a helper message.

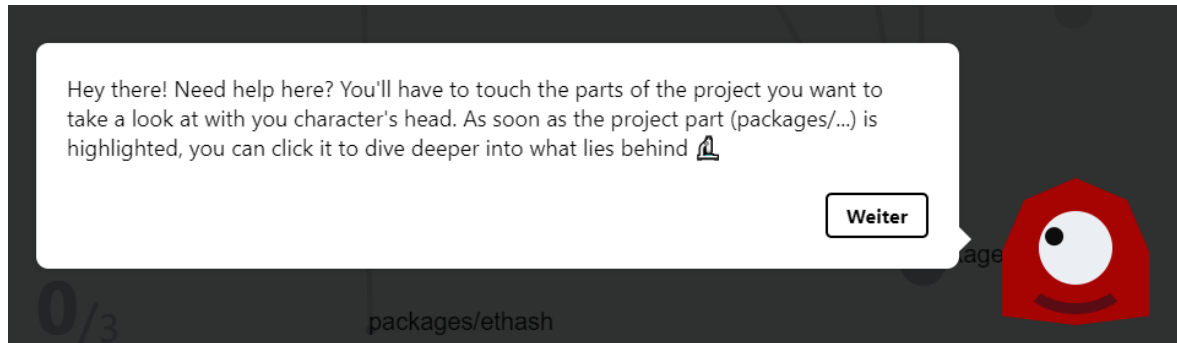


Figure 9: Helper companion within the technology probe

Within that figure, attempts on making the companion having more personality are also visible. The companion’s pupil follows the player’s cursor to make it feel more connected to the player’s actions for example. In addition to that the companion should be seem more approachable with the friendly facial features, as well as the usage of a friendly tone of text together with the usage of emoticons. If these attributes actually achieve the goal of a companion with personality, is subject to discussion based on the probing results. For the implementation details of the companion, as well as the full set of companion messages, refer to the probe codebase³⁶³⁷³⁸.

This concludes the user-facing implementation of the technology probe. The actual generation and size of the scenes as well as the game objects is based on the underlying project’s metadata and source code. How the data is programmatically read, what data is used and how it is integrated into the technology probe, is therefore subject of discussion within the next chapter.

4.3.2 Game World Generation

In order to generate a game world from a project’s source code and metadata multiple distinct levels of tasks are necessary. To be more specific, at first the data that is needed to generate the world has to be defined, based on what outcomes are wished for and what the underlying data can provide. After that, based on the desired data, the location of that data (or the raw data necessary to then aggregate what is needed) has to be identified. The data can then be programmatically fetched from these locations and aggregated into a format that

³⁶See suppl. repository at path: `./technology-probe/src/ui/companion.ts`

³⁷See suppl. repository at path: `./technology-probe/src/ui/intro.ts`

³⁸See suppl. repository at path: `./technology-probe/src/scenes/DetailScene.ts`

the web page containing the user-facing parts can handle. Based on that data, ultimately the game world as well as the objects in it can be generated.

As described, the first step to take, is to clarify which data is needed. The most important question to answer here, is: What is the desired outcome? In the case of this technology probe, the data is needed to generate game objects and the overall game world of the technology probe. Therefore it has to have properties that make it usable for such a generation, like a numeric value that can be used to generate a game object with a distinct size. Additionally contextual information important for the newcomers can be shown in the mentioned info messages. Based on these constraints the data needed for the different applications throughout the probe is aggregated in *Table 2*.

Application	Data Needed
(1) Subproject Objects (<i>Overview Scene</i>)	Name, Size & Connections to other sub-projects
(2) Discoverable Objects (<i>Detail Scene</i>)	Size of the object
(3) Contributor Messages (<i>Detail Scene</i>)	(User-)Name, Link to <i>Github</i> profile, Last activities within the project, Number of contributions
(4) 3rd-party Package Messages (<i>Detail Scene</i>)	Name, Version Number, Link to package website
(5) Potential Issue Messages (<i>Detail Scene</i>)	Type of Issue, Issue Location, Link to Issue Location, Size of the Issue

Table 2: Data needed to generate the technology probe game world

In order to match the contextual information to the in-game world, the size should match the size of the respective subprojects or potential issues. As there is no directly readable size attribute, an estimation of the size in relation to the other subprojects and issues has to be found. Concerning the subprojects the most evident correlation would be to the disk space occupied by respective directory. Concerning the contextual information, some of it can be read directly from the `package.json` files within the subprojects where *Node.js* projects store information about all of the dependencies, both within the monorepo and to 3rd-party packages. Other information, like contributor profiles or their last activities in contrast have to be requested from *Github*.

Thus, the data can be collected from the following locations. Concerning *application (1)* (refer to *Table 2* for these applications), the name of the subproject as well as its occupied disk space (=size) and the connections to other subprojects can be read directly from their respective `package.json` file. A `package.json` file does have a standardized schema and includes all 3rd-party packages used in two properties, called `dependencies` and

devDependencies. Connections within the monorepo are also contained within those properties and marked with a prefix (in this case `@ethereumjs/`).

For application (2) the only attribute needed is its size, as the discoverable objects are randomly placed in the *detail scenes* and are not discernible by type. The size itself depends on the type of object, though, as contributors do have other attributes than 3rd-party-packages for example. Therefore, the size of the respective shape has to be generated based on a different data attribute.

Regarding the objects based on contributors (application (3)), this is going to be the number of contributions, as it is an easily accessible number, available through the *Github Application Programming Interface (API)*³⁹. Further, it also is comparable between the different types of contributors and increases the visual weight of the most important contributors. The other data points needed (name, link to profile, last activities) can also be easily fetched from the aforementioned *Github API*.

Concerning application (4), the 3rd-party-package information messages, the data for the package name and currently installed version can be sourced directly from the respective `package.json`. The link to the package website is not as easy to uncover, as it is not documented within the project itself. As a workaround, the link to the package subsite within the Node Package Manager (NPM) ecosystem is created through combining the package name with the base Uniform Resource Locator (URL) of the NPM website⁴⁰.

Lastly the data concerning application (5) has to be collected and aggregated into what is needed. To do this, a certain type of issue has to be decided upon. There are multiple possible issues especially in large codebases. In order to keep implementation efforts at a manageable level, only one kind of issue is going to be scanned for in the project. A possible issue, that is easily discoverable from the source project's files alone, are large files. While not necessarily directly connected to badly written code, such large files can be hard to understand and messy [98] for newcomers. Therefore, the identification of such files could present an angle for communication with existing contributors, or creating issues for possible refactoring within the source project's repository – possibly as way to introduce themselves into a project. With the choice on the type of issue out of the way, the needed data (file location, link to the file location and the size of the file) can be gathered directly from the underlying project.

After clarifying, where the data that is needed can be collected from, the actual collection has to happen. Based on the clarification there are two distinct locations, where data has to be

³⁹<https://docs.github.com/en/rest>, accessed on 14th of August 2021

⁴⁰https://www.npmjs.com/package/PACKAGE_NAME

fetched from. Firstly, the codebase's files and directories and secondly *Github's public API*. The simplest way to access that information is, to do it locally through a script, that programmatically reads from the file system and accesses the necessary API. To stay in the *Javascript* ecosystem, the script is created with *Node.js* and can be found within the technology probe's source code as well⁴¹. This script can be executed manually and produces an output file containing all the necessary information needed to generate the game world. This file can also be found in the same repository⁴². This file contains all of that information in a file format (.json) readable by the technology probe's source code, in a format that can be easily used for generating the game world. It includes data that is already aggregated into a format that can be used as-is to generate all the game objects, including the size of the shapes, the textual content shown in the info messages, and the links necessary to lead the player to the actual contributors, packages, and more. This is then directly used within the context of the webpage to generate the game objects. Again, the implementation details can be found in the respective repository⁴³.

4.4 Data Collection

In addition to the user-facing implementation and the creation of the game world, user-generated data has to be collected. This can then be used to draw conclusions from the probe as a whole and to ultimately answer the research questions. As already mentioned in the methodology chapter, user-generated data is collected in four different ways: **Capturing demographic information (1), logging participant interactions (2), questions on the underlying project (3) & questions on the research topic (4)**. For all of these ways, the specific questions – or logging events for that matter – have to be defined and implemented.

Starting with the demographic information, this should just give some additional contextual information to the participant's background. The primary goal is to research how developers experience playful elements in software development projects unbeknownst to them. Thus, questions on the professional background of the participants are crucial. The participants are also given the option to give their name and age, if they so desire. That concludes the small set of demographic questions to be asked. As said, these are just to give some additional context for the result argumentation and to make sure that the participant is part of the

⁴¹See suppl. repository at path: `./technology-probe/scripts/get-metadata.js`

⁴²See suppl. repository at path: `./technology-probe/metadata/project.json`

⁴³see page 4 for the accompanying repository URL

target group. Because there are not going to be any generalizeable or representative conclusions, no additional demographic information is asked for. The final set of input from the participant at this stage is presented in *Appendix E*.

Regarding the second way of collecting data – logging user interactions within the probe – additional considerations have to be made. More specifically, the types of the interactions have to be considered and how they are actually logged within the probe. Firstly, a structure has to be defined on how the single events are logged and distinguish the events from each other. The distinction itself is done through a pre-defined two-character code, so that it can easily be aggregated and analyzed if needed. The full table of all the codes used throughout the probe can be found in *Appendix F*. To give context to the singular events, when logging these, timestamps are included to keep track of when and in what sequence these events happened. Additionally, there is the possibility to add custom text to these events if so needed. Within the probe’s codebase a function was created (see the respective file for the implementation details⁴⁴, making it possible to track events from within the code. These are then persisted through Google’s Firebase ⁴⁵ service, where for each participant the answers to all questions and the logged events are stored for later analysis.

The third way of collecting data, is concerned with getting a sense on how the data about the project translated into the participant’s understanding of those. Therefore, it should consist of question on what was presented regarding the underlying projects. These questions were based on the metadata collected for the generation of the game world and directly target what was presented. The final set of questions can be found in *Appendix E*. Question (1) is about how vividly the participant remembers the contributors shown, as they are the prime social links of a project. Question (2) is added as a way of finding out if the participants connected the subproject size to the size of the shape displayed in the overview scene. The next question (3) is included as a way to find out if the participant recognizes what the lines between the subproject game objects are trying to visualize and how memorable the overall visualization in the *overview scene* was. Question (4) targets the 3rd-party-packages and if the participant remembers some of the ones that were discovered. The last question (5) is targeted towards how participants viewed the last kind of revealable information, the large files that were identified within the game world generation procedure. This last question also includes the participant’s opinion on how they would rate the issue of these large files in terms of graveness of the issue. While this answer is not strictly confined to the underlying

⁴⁴Source code file within the accompanying repository at: `./technology-probe/src/logger.ts`

⁴⁵<https://firebase.google.com/>, accessed on 14th of August 2021

project it is very much connected to the topic of the question. Additionally, it transitions into the questions on the research topic as a whole.

These questions on the overall research topic are also included in *Appendix E*. In the following, for each of the questions a reasoning is given on why it was included. Question (1) tries to investigate if the participants themselves subjectively feel like they have learnt something on the underlying project. To make sure to get more than just a positive or negative reply, within the question it is explicitly mentioned, that the participant should give an answer on what was learned. Additionally, if the participant did not feel like something was learned, an explicit enquiry to what should have been included is added to the question. Question (2) tries to expand from the learning or onboarding aspect into the overall subjective experience of the participant exposed to the probe. The question is asked in an open-ended manner in order to get qualitative data based on the participant's experiences throughout the probe. Specifically mentioning parts that were liked or disliked by the participant is also encouraged through the wording of the question.

The following question (3) focuses on a single feature of the probe, that might be of value also in contexts other than the technology probe – the companion and how the participant experienced it. It also acknowledges that the companion might only have been visible as a status indicator not as a helping feature.

Question (4) opens up again, more towards how the participants would describe their stance towards an interactive visualization (as is the case with this technology probe) for software development onboarding purposes. Additionally, it asks for input on projects that might be adequate for such visualizations and on which projects the participant would want to try them out. For, if the participant expresses that they cannot imagine themselves using something like that, a direct query on what they would use instead is included.

Question (5) completely moves away from playful elements in onboarding and tries to get a personal perspective on how each of the participants usually approach onboarding. Specific cues are given to give the participants a clue on what interesting elements of the onboarding process are.

The question after (6) then is more of an optional question, targeted at developers working on open source projects specifically. As the onboarding in such projects is different from a local on-site team, it is included as a way to gather information about how the nature of the project changes how an onboarding process is done at a personal level.

Question (7) then focuses on the explorative nature of the research done within this thesis. It tries to gather input from the *experts* on the topic, the developers, the participants on how

they would integrate playful elements in an onboarding context.

Question (8) builds upon that and asks for two things. Firstly, the general stance towards the usage of playful elements in onboarding processes in order to gather information about developers' stance towards playfulness in their work environment. Secondly, the nature of the project is also included as a way to gather data on how participants would perceive playful elements in different contexts (namely working on open source software vs. commercial products).

In the last formulated question (9) the participants are asked for game mechanics familiar to them, that they would deem valuable as playful elements in onboarding processes. While this is a larger task for the participants, input on that question is very valuable, as it further explores ways of using play and playfulness in onboarding contexts.

Finally, question (10) is included as a space, where the participants can give additional feedback independently of the preceding questions.

Overall, all the questions are open-ended where possible, to gather as much qualitative input as possible. Due to the asynchronous and remote nature of the probe, there is no possibility to immediately query the participant on what was said before, though. Due to this downside of the approach, these queries are included within the question, where possible. Concerning the technical implementation, the questions are logged to *Firebase* and added to the participant's data, together with the demographic data and logged events. How the questions are included within the technology probe as a whole is laid out in detail in chapter 4.6.

4.5 Participant Selection

After clarifying how the technology probe is implemented, how the game world is generated and how the qualitative data is gathered only two aspects of the probe are left before the study can be executed. Firstly, all of the single parts of the technology probe have to be put into a distinct sequence, so that the participants are introduced to the study, can give their consent, go through the probe and finally answer the questions outlined before. This is going to be outlined in the next chapter. Secondly, that sequence has to be gone through by a set of participants, that have to be selected based on a set of requirements.

The single most important requirement is that the probe is executed by participants that are part of the context that should be investigated. In this case, these are software developers of any kind. There is no need to limit the selected participants to a certain group or level of experience, as the topic of onboarding is something every kind of developer is exposed to some degree. On the contrary, to get a wide variety of different responses it is beneficial to

have a very diverse set of participants (concerning age, experience, and more). Thus, the only requirement concerning the participants is going to be that they are software developers, either professionally, on a leisurely basis or nascent ones just starting out in the field. As the study is going to be held remotely and asynchronously, the participants do not have to agree onto a fixed data and time, but can choose for themselves when they want to take part within the probe.

Concerning the number of participants needed, there is no definite amount of participants needed to reach a representative conclusion. Rather, the number of participants should be enough, that an amount of data is generated that produces a big enough foundation to answer the research questions. This is not tied to a set number of participants, but to the data generated by each participant. To get answers that are diverse enough, a minimum number of participants should be set – in this case five developers. If, after reaching that number of participants in the probe, not enough data to draw conclusions from is generated, additional participants are going to be asked to take part in the study.

4.6 The Final Technology Probe

At this point every part of the probe, the constraints around it and the desired participants are defined. As mentioned before, the only step left at this point is to create a coherent probe that includes all of the carefully created parts in a structured manner. To recapitulate, these parts are: **An introduction to the study, a consent and data protection form, the interactive visualization of the @ethereumjs project, the demographic questions, the questions on the underlying project & the questions on the research topic.** All of that has to be combined into an easily accessible technology probe accessible through a browser on a webpage. Together with introductory messages and further information for the participant, the final structure of the probe is as follows:

1. Introduction message to inform the participant on the study, the goal of the study, and the overall procedure
2. Consent & Data Protection to follow the research ethics guidelines
3. Demographic questions
4. An introduction to the interactive part of the probe
5. Beginning of the interactive part within the probe overview scene
6. Revealing important parts of the project within the probe detail scenes
7. Questions on the underlying project
8. Questions on the research topic

9. Outro message

For all of these steps a separate subpage has been created that can be seen in detail in *Appendix G*. The implementation of the view logic to cycle through these steps and the elements and styles of these subpages has been done outside of the interactive *p5.js* application with just *HTML*, *CSS*, and *Javascript (Typescript)*. Details of the implementation can be found mainly in the two respective files of the technology probe codebase⁴⁶⁴⁷⁴⁸. The companion, that was described in detail in chapter 4.3.1 is ultimately included on the lower right of the interactive part of the probe. In addition to that, a counter was added to the technology probe steps 5 & 6, so that participants get a sense of their progress within the probes. Additionally, the exploration of the subprojects is limited to three, to limit the amount of time needed to complete these steps of the probe. This concludes the documentation of the design and implementation of the research done within this thesis. The next chapter now focuses on the results & conclusions drawn from that research.

⁴⁶See suppl. repository at path: `./technology-probe/src/ui/intro.ts`

⁴⁷See suppl. repository at path: `./technology-probe/index.html`

⁴⁸See suppl. repository at path: `./technology-probe/styles.scss`

5 Results

With the study executed and analyzed – using the methods described in the methodology chapter and following the blueprint laid out in the research design chapter – the results can now be elaborated on in detail and connected to the research questions of this thesis. This elaboration starts with recapitulating and aggregating the final set of playful design implications discovered in literature. After that the results of the study are described in detail as well. This includes the final number of participants, a contextual description of the participants, and their qualitatively analyzed input supported with the answers on the underlying project and the data logged throughout the probe execution. Finally, these results are set into the context of the research questions, so that these can be answered with what was researched throughout the whole thesis.

5.1 Playful Design Guidelines

The foundation for the playful design guidelines were discovered through the state-of-the-art literature research and the literature research on the theoretical grounding of this thesis. The actual guidelines were ultimately created as the first step towards the probe implementation. An excerpt of these implications was mentioned before, and the full set of implications can be found in the respective diagram in *Appendix B*. The final set of playful design implications therefore can be summarized as follows:

- Use serendipitous effects throughout the context
- Let the player create freely within a space
- Let the player change the context and visualize the effects
- Let the player manipulate and destroy parts of the context and visualize the effects
- Intentionally let the player see through the cracks of the game world
- Create a game world from contextual data
- Humanize the context and generate communication through it
- Create a stage for the player, where they can explore without too much guidance
- Use uncertainty & do not reveal everything to the player
- Utilize playful elements already present in the context
- Allow for informal and personal communication with the player
- Purposefully create mysterious elements

Some of these implications resemble very closely what was discovered in existing literature, some are aggregations of multiple sources done by me. For the specific literature this is based

upon, refer to the respective diagram in *Appendix B* and the reasoning in chapter 2.4. This set of design guidelines is one part of the contributions of this thesis as a whole. It informed the creation of the technology probe outlined in the research design chapter, but could also serve as guidelines for the future creation of other playful designs.

5.2 Study Results

With the recapitulation of these guidelines as the foundation of the technology probe, the focus can now shift towards the results of that technology probe. This probe is based on the methods mentioned and discussed about in the methodology chapter and was executed as described within the research design. At first there is going to be an overview of the final set of participants. This is enriched with some general aggregate data computed from logging the interactions during probe execution. After that the results of the interim goals during the qualitative content analysis are presented. Ultimately, an attempt at answering the research questions through interpreting and presenting the results of that analysis is made.

Before that can be done, as mentioned, the set of participants has to be presented. As described in both the chapter on methodology and research design, the only requirement towards the participants is that they are a software developer of some sorts. Therefore, two companies that employ software developers were initially contacted and asked to share the URL to the technology probe with their software developers. This resulted in five initial participants. As the resulting data was not sufficiently diverse, four additional developers were contacted directly via e-mail and also asked to share the probe with their colleagues. Ultimately, nine participants executed the technology probe, with eight of them finishing the interactive part and five finishing the interactive part as well as providing answers to all the questions. The participants were aged 24 to 36, and the participants professional experience varied from zero to more than 10 years. Professions of the participants also varied in detail, with web and software developers, data scientists and others. The final selection with the specific information on each participant and a participant code, that is going to be used going forward can be seen in *Appendix H*.

Overall these eight participants went through the interactive part of the probe in times between 2 minutes 18 seconds and 12 minutes 50 seconds. With the exception of one outlier, around 120 to 180 interactions were needed to discover all the revealables generated for the probes. While these aggregate numbers do not carry much meaning by themselves, they (and the single interactions log data points they are based upon) can be used to add contextual information to the qualitative data gathered throughout the study. An overview of these

aggregations for each participant can be found in *Appendix I*.

Concerning the content analysis of the qualitative data, it was executed as described in chapter 3.4. In short, there are four phases to go through with the chosen approach: *Selecting material, creating a category system and coding the content with the specified categories*. As mentioned earlier, selecting the material is trivial in the case of this study. Where additional work was done, is in developing a category system that then is used for coding the material, as described by Schreier [89, p. 174]. To develop such a system it is crucial to clarify what should be achieved by the analysis. In this case, the ultimate goal is to answer the research questions with the data that was generated. Thus, these categories should serve the goal of coding the material with the research questions in mind. This can be achieved by deductively creating the categories from the research questions and the literature they are based upon. The result of this procedure can be seen in *Table 3*.

Research Question	Categories
(RQ1) How and which theories & concepts from research on play can be used in the onboarding process of software development projects and how are developers experiencing them?	Ideas on playful elements for onboarding
(RQ2) Which playful themes could lend themselves well to evolve software development projects into a spaces of play?	Experience towards playful probe features
(RQ3) What are the technical, social and personal intricacies of an onboarding context in software development and how do they influence the onboarding experience?	Social Onboarding (Aspects), Technical Onboarding (Aspects), Organizational Onboarding (Aspects), Ideas on Onboarding Elements
(RQ4) How can the identified themes be applied in actual onboarding settings?	Improvements for implemented features
(RQ5) How do different kinds of software developers experience playful aspects within an onboarding context?	Attitude towards Play in Onboarding, Contextual Considerations

Table 3: Content Analysis – Category System developed from the research questions

For each of the research questions at least one category has been created, aiming to synthesize valuable data that can be used to answer these questions. If during the analysis of the coding categories were deemed not concise enough for important units of content, sub-categories were inductively created. More specifically, the category of *ideas on onboarding elements* was added to create a distinction on feature-oriented input by the participants. Additionally, a category on *contextual considerations* has been created, as there has been multiple instances input on how the approach on playfulness changes depending on the onboarding context.

These categories as a whole then were used to code the content. Where these codes were not enough to create a meaningful unit of content, contextual units were used. As described in chapter 3.4, these contextual units can be supportive data from the logs as well as the playful features themselves. With that information distinct units of text can be coded and set into context and ultimately interpreted in relation to the research questions.

The actual coding process that came after, was two-fold. At first, units of text that contained important content was marked. As a second step, these units of text were then collected and coded using the set of categories defined beforehand. After creating the two additional, inductive categories, the units were examined again, to adapt categories where needed. The result of this process, with every single coded unit of text together with contextual units – where applicable – can be seen in *Appendix J*. Throughout the results chapter, if a coding unit is referred to, it is done so by an auto-generated number also present in the table within the respective *Appendix J*.

Together with all of the work that has been executed prior – researching the origins of play, contemporary views, creating design guidelines, implementing features, creating a technology probe – this finally allows for careful interpretation of everything that was gathered up until this point.

(RQ2) Which playful themes could lend themselves well to evolve software development projects into a spaces of play?

Concerning this research question much of the effort has already been expended. As described in chapter 3.1, the playful themes were discovered through literature research and careful condensing of what was found into foundations for the playful design guidelines. Thus, this can be seen as the answer to the first part of this question. If they lend themselves well in the context of onboarding onto software development projects is unanswered, though. As was mentioned in chapter 3.4 as well, a generalizeable and reliable answer on effectiveness is out of scope in this thesis. Rather, initial feedback on the subset of playful features implemented has been gathered and can serve as guidance for future applications of play in onboarding processes.

Concerning the theme of creating an intentionally ambiguous game world for example there were mixed irritated experiences reported by the participants. P1 mentioned here: ‘I was a bit confused because I didn’t know what to do when I reached the first blank slide’⁴⁹.

Regarding P1 this was also supported by the logged interactions, where the participant did

⁴⁹Coding Unit Nr. 1 in *Appendix J*

not perform any action at all, until the companion showed the help message⁵⁰. Multiple other participants mentioned that as well. P3 also mentioned that it was not clear how the reveal process within the detail scene should begin and that he did not know how to start in that scene⁵¹, which also is supported by this participant's logs⁵². P8 mentioned some problems with the detail scene as well, mentioning 'I needed some time and the help box to understand, that I was supposed to click somewhere to reveal the information'⁵³. Overall it could be assumed, that the large amount of ambiguity, with completely hidden objects may have gone too far. This is even more important when using those elements as a way to gain knowledge. Here small indicators, or evolving objects that initially are shown in a game world, might have resulted in better understanding.

Here the companion feature was very helpful for the participants, though. P1 for example mentioned that 'helper on the bottom was quick to assist, though'⁵⁴ and that it 'came to help always at the right time. Good guy!'⁵⁵. This was also mentioned by P3, albeit in a less personal tone⁵⁶ and with the notion, that there was no interaction other than the help messages⁵⁷ (this was also mentioned by P6⁵⁸). Overall though, the experience towards the companion was very positive and could be a theme worth exploring further in this context. The reveal mechanic – spawning bubbles to reveal information – on the other hand was perceived more mixed. P8 mentioned that it was fun at first, but in the end it got repetitive⁵⁹. P8 also reported problems catching the revealable objects⁶⁰, which can be traced to the logs as well⁶¹. Albeit the technology probe should be kept very simple according to the literature on it, additional or changing mechanics could have helped here.

Other important notations were targeted towards the overview scene. Both P8 and P6 mentioned that the randomly generated overview was pretty chaotic and therefore the structure of the subprojects were hard to see⁶². A more structured visualization could have helped here, although P9 specifically pointed out, that it was helpful to see objects for all the

⁵⁰see lines 10-63 in suppl. repository at path: `./supplementary-material/raw-probe-data/technology-probe-logs.json`

⁵¹Coding Unit Nr. **44** in *Appendix J*

⁵²see lines 9040-9090 in suppl. repository at path: `./supplementary-material/raw-probe-data/technology-probe-logs.json`

⁵³Coding Unit Nr. **91** in *Appendix J*

⁵⁴Coding Unit Nr. **14** in *Appendix J*

⁵⁵Coding Unit Nr. **17** in *Appendix J*

⁵⁶Coding Unit Nr. **45** in *Appendix J*

⁵⁷Coding Unit Nr. **48** in *Appendix J*

⁵⁸Coding Unit Nr. **71** in *Appendix J*

⁵⁹Coding Unit Nr. **93** in *Appendix J*

⁶⁰Coding Unit Nr. **93** in *Appendix J*

⁶¹see column *Contextual Unit (Logs)* Nr. 93 *Appendix J*

⁶²Coding Units Nr. **64, 94** in *Appendix J*

subprojects⁶³. Mostly positive feedback was gathered towards the aspects of the underlying project⁶⁴. It has to be mentioned, though, that especially P1 had mixed feedback⁶⁵.

Important to note here, might be that P1 – in contrast to all other participants – mentioned that contributors ‘don’t really help me to get started with a project I guess’⁶⁶.

Overall the study reinforced some playful themes, especially the companion with personal communication and help towards the participants. Others, like the extremely ambiguous initial detail scene, irritated more than it helped. Crucial to mention here is that with this study setup no definitive assertion can be made on the effectiveness of these features, it only explores personal subjective experiences of the participants – to possibly inform such studies on subsets of playful themes in the future.

(RQ3) What are the technical, social and personal intricacies of an onboarding context in software development and how do they influence the onboarding experience?

The approach to answering RQ3 is very similar to the previously discussed research question. Regarding this question, a considerable amount of literature exists and was discussed during the literature research part of this thesis. To summarize, there are different schools of approaches towards software development onboarding: Psychology-centered, process-centered & developer-centered approaches [42]. These all focus on different aspects of the onboarding process. Technical intricacies (psychology-centered approaches) can be program comprehension and programming (or similar purely technical) skills. Social – and organizational – intricacies include the project setup as a whole (remote vs. on-site, open-source vs. commercial), but also the project members, experiences of colleagues working on the same project, availability of colleagues or organizational and hierarchical structures behind a project. Lastly, personal intricacies are overlapping with both technical and social challenges. These include personal approaches and processes towards onboarding, like information seeking in projects (no matter if it is organizational or technical information) or feature locating within source code. In my opinion creating a solution that spans every aspect of onboarding is near impossible to achieve as it is such a broad topic. Thus, when creating a supportive digital solution, focusing on a distinct approach is crucial, e.g. by deciding on creating something that supports program comprehension or deciding on improving in-project

⁶³Coding Unit Nr. **116** in *Appendix J*

⁶⁴Coding Units Nr. **37, 38, 41, 42, 88, 118** in *Appendix J*

⁶⁵Coding Units Nr. **4, 12** in *Appendix J*

⁶⁶Coding Unit Nr. **3** in *Appendix J*

communication, as vastly different intricacies are present depending on the area of interest. For each of these area of interests research already exists, as mentioned in chapter 2. Thus, an extension of the body of work is not necessary in the case of this question. Nonetheless, the literature discussed and knowledge gained is and was valuable as a foundation for the other parts of the thesis. Where it is important to gain new knowledge is to contextualize the answers the participants provided. This is necessary to understand how different personal approaches towards onboarding might influence the attitude towards playfulness in those approaches.

Concerning this research question, data was gathered on three mentioned different aspects (social, technical, organizational) and ideas from the participants towards aspects of onboarding that are of importance. Here the participants reported very different attitudes towards information in onboarding. P1 for example, as mentioned before, deemphasized the importance of contributors in onboarding⁶⁷. Although it has to be noted, that in on-site projects P1 mentions that social contact and direct communication is important⁶⁸. As P1 mentions having not a lot of onboarding occasions and 0-1 years of experience as a developer, this can be set in a context, where P1 might either see contributors not as direct colleagues or has more experience with on-site projects. Concerning these on-site projects, there is consensus between the participants that social links might be the most important in onboarding⁶⁹.

A lot of technical onboarding considerations were mentioned by the participants as well. The participants reported mostly similar technical approaches to projects. P3, P8 and P9 all mentioned starting at the top-level of a project and then going into the details of single files and features of these projects⁷⁰. An important aspect of this initial approach is the access to the project itself, as for example P6 mentioned: ‘I try to get access to everything that I need (git, deployment server, ...)’⁷¹. For P6, the ‘first step is always the documentation, if there are problems I google or search through the issue list’⁷². This statement reinforces the importance of existing information on the project, be it within documentation or in 3rd-party forums. Here the project maintainers themselves can allow for a smoother onboarding experience by providing that – if these are provided in a coherent manner, that kind of structure can be used within playful elements. Concerning the tools that are used throughout

⁶⁷Coding Unit Nr. **3, 8** in *Appendix J*

⁶⁸Coding Unit Nr. **23** in *Appendix J*

⁶⁹Coding Units Nr. **53, 59, 76, 98, 123** in *Appendix J*

⁷⁰Coding Units Nr. **51, 56, 99, 122** in *Appendix J*

⁷¹Coding Unit Nr. **75** in *Appendix J*

⁷²Coding Unit Nr. **80** in *Appendix J*

the onboarding process, mostly working directly within the IDE was mentioned⁷³. A notable addition to that came from P8, was that the initial code navigation happens on Github (for projects hosted on Github at least)⁷⁴.

Regarding the organizational aspects, not as much input was given. There are two notable exceptions, though. P3 mentioned that it is very helpful to have a structured and well-organized issue list helps in identifying possible issues to start working on⁷⁵. P6 also brought up the fact that additional communication with non-developers might be worthwhile⁷⁶. This – arguably also social aspect – expands the onboarding further into non-technical areas, which have not been the focus up until this point.

To get back to answering the research question as a whole, this wide variety of aspects has to be contextualized. Overall, there are different areas in an onboarding process with each having different requirements and problems inherent to them. Thus, these changing requirements, depending on what is to be achieved, have to be kept closely in mind. In the case of this thesis, this meant for example, that the technical onboarding aspects were pronounced and considered in detail. For future approaches, focusing on the social aspects a different approach might be needed on the other hand.

(RQ4) How can the identified themes be applied in actual onboarding settings?

For a subset of themes, this question can be considered answered already, as a possible application has been done as a part of the implementation of the technology probe.

Concerning the playful themes and guidelines that were not implemented, possible applications can be subject to future research into that area. In addition to the documentation of the implementation process and thus the answer towards how to apply some of the themes, feedback on these applications was gathered throughout the study. These can inform how those applications could be improved or extended upon.

To be precise, broad input was provided on how existing features could be improved upon incrementally, but also where the current approach was experienced as lacking. Starting with the visualization of the project as a whole P8 mentioned an improved application through suggesting, that ‘there could be some moving dots from module to module, depending on how often the module gets imported into another module’⁷⁷ and to adapt the visualization to

⁷³Coding Units Nr. **24, 79, 100, 122** in *Appendix J*

⁷⁴Coding Unit Nr. **99** in *Appendix J*

⁷⁵Coding Unit Nr. **55** in *Appendix J*

⁷⁶Coding Unit Nr. **78** in *Appendix J*

⁷⁷Coding Unit Nr. **101** in *Appendix J*

grasp the importance of these modules⁷⁸. This could achieve a more detailed understanding of the underlying project through leveraging more detailed data from within the project itself. Integrating more data within these visualizations was also mentioned by other participants. P3 mentioned that non-technical information should have been included as well, such as the overall functions of the project itself⁷⁹. P6 also proposed presenting more information within the visualization, by proposing a visualization of the project organization structure together with suggestions of who to communicate with⁸⁰ and a stronger connection to the project's source⁸¹. As of smaller improvements to the implemented features, P1 mentioned adding more images and icons to the information on revealable objects⁸² and described how the reveal mechanic could be made more enticing, by increasing depth perception through using different colors⁸³.

Overall, most of the participants interacted closely with the implemented features and therefore the application of playful themes. Only P6 went through the probe very fast (as can be seen in *Appendix I*) and did provide not that much feedback on game mechanic intricacies or detailed tasks, rather the feedback was targeted more towards the probe as a whole. Nonetheless, on all of the levels of detail, valuable input was gained on possible improvements to applications of play.

(RQ5) How do different kinds of software developers experience playful aspects within an onboarding context?

The last of the (sub-)research questions is then specifically crafted to foray into a non-existing body of research – the experience and attitude of developers towards playful elements in onboarding processes. The questions created for the probe mirror that and try to generate data on exactly that aspect of the research topic. The study ultimately succeeded in doing so and the participants brought forth many interesting aspects to this topic, that are going to be laid out in detail at this point. This allows for documenting different experiences to what was built by a set of participants with different experiences and professions.

P1 had a very clear opinion on playful elements for themselves and stated ‘this kind of visualization isn’t the right thing to learn about a coding project. At least not for me.’⁸⁴ P1

⁷⁸Coding Unit Nr. **97** in *Appendix J*

⁷⁹Coding Unit Nr. **43** in *Appendix J*

⁸⁰Coding Unit Nr. **82** in *Appendix J*

⁸¹Coding Unit Nr. **69** in *Appendix J*

⁸²Coding Unit Nr. **10** in *Appendix J*

⁸³Coding Unit Nr. **16** in *Appendix J*

⁸⁴Coding Unit Nr. **20** in *Appendix J*

underlines that by giving an opinion on playful elements in onboarding processes as a whole. More specifically P1 mentions urgency and time available to get to know a new project and states that in time-sensitive situations playful elements hinder effective onboarding⁸⁵. P1 even describes it as potentially ‘frustrating’⁸⁶. What P1 also describes is the need for the right time and mindset necessary for a playful exploration of a project, where parts of a project are explored that might not be connected directly to the work done at the moment⁸⁷. An important aspect that P1 also mentions is that playful exploration should be done at a pace individual to the respective developer⁸⁸.

P3’s attitude towards playful elements stands in contrast to P1’s statements – at least regarding the general opinion on those playful elements. P3 specifically brings up that a visualization coupled with playful exploration techniques might be able to provide a good *first* overview of a project⁸⁹.

P6 shares these sentiments with specifically pointing out that the playful elements and the exploration could provide a benefit to get an overview of software development projects⁹⁰. P6 acknowledges that this is not true at all times, though, by mentioning the inefficiency of such an approach in day-to-day work⁹¹ and that technical onboarding might better be done through IDEs and the such⁹². P8 gives similar feedback, with mentioning the possible inefficiencies – largely caused by the *randomness* of the revealable objects – of a playful approach⁹³. However, P8 also found aspects of the approach helpful, like pointing out potential issues of the underlying project⁹⁴, although that statement is not directly related to a playful approach, rather on the data extracted from the underlying project. P9 points out similar sentiments, albeit targeted more towards projects where getting an overview from the outside just by investigating the source code files might be hard to achieve⁹⁵.

To summarize these findings, there are mixed results on developer experience and attitude towards playful elements in onboarding. While there are statements towards the inefficiency of playful elements, the usage of these elements for getting initial overviews and exploring parts of the project that are not directly worked with, was described positively. Overall the

⁸⁵Coding Unit Nr. **18, 19, 30** in *Appendix J*

⁸⁶Coding Unit Nr. **19** in *Appendix J*

⁸⁷Coding Unit Nr. **31** in *Appendix J*

⁸⁸Coding Unit Nr. **20** in *Appendix J*

⁸⁹Coding Unit Nr. **50, 60** in *Appendix J*

⁹⁰Coding Unit Nr. **72** in *Appendix J*

⁹¹Coding Unit Nr. **83** in *Appendix J*

⁹²Coding Unit Nr. **74** in *Appendix J*

⁹³Coding Unit Nr. **96** in *Appendix J*

⁹⁴Coding Unit Nr. **89** in *Appendix J*

⁹⁵Coding Unit Nr. **121** in *Appendix J*

participants reported no fundamental resentment towards play in onboarding processes, but mentioned important considerations and possible downsides of such approaches. Some of those important considerations are connected to the context in which onboarding takes pace. Aspects of this were mentioned by multiple participants based on the question of playful elements within a work environment and open-source projects. P3 and P8 for example mention that in on-site working environments the access to other project members and experts can be significantly better. Thus direct in-person communication – which is preferable for the participants – is easier to achieve. Therefore, both P3 and P8 mention that an automated playful introduction should be more valuable when this in-person communication is not feasible⁹⁶. P6 and P8 also mention the motivation behind working on open source projects (‘out of interest’⁹⁷ or in a ‘self-motivated’⁹⁸ way), which could increase the openness towards exploring playfully to learn and gain knowledge on a personal level – rather than focusing on only what is needed to work on specific tasks. P6 also points out that playful approaches might be valuable for young developers or people starting out with development⁹⁹. Lastly P9, adds a another contextual consideration not strictly confined to open source projects but important nonetheless. P9 mentions that in large software projects that are hard to grasp, playful elements and exploration techniques could provide value¹⁰⁰. P9 also specifically points out that this could be helpful for projects built with programming languages the developer is not proficient in, but wants to learn for themselves¹⁰¹.

(RQ1 – Main RQ) How and which theories & concepts from research on play can be used in the onboarding process of software development projects and how are developers experiencing them?

After answering all the sub-research questions an answer for the overall main research question has to be crafted. Such an answer requires considering the answers to all the previous RQs, additional information from literature, but also all the explorative information generated throughout the study.

In order to use that information on an answer to this question the question has to be split up into its distinct parts: The playful theories & concepts on one hand and the developers experience towards them. Concerning the playful theories & concepts in onboarding processes

⁹⁶compare Coding Units Nr. **61, 62, 107, 109** in *Appendix J*

⁹⁷Coding Unit Nr. **84** in *Appendix J*

⁹⁸Coding Unit Nr. **108** in *Appendix J*

⁹⁹Coding Unit Nr. **85** in *Appendix J*

¹⁰⁰Coding Units Nr. **121, 124** in *Appendix J*

¹⁰¹Coding Unit Nr. **44** in *Appendix J*

a lot of results have already been generated through the explorative research that happened. This is now enriched with the analyzed study data, where possible. Regarding the second part of the question, this mostly was answered through RQ5.

As said, a lot of different possibilities of which playful theories & concepts could be applied in an onboarding process has already been explored, which is described in detail in the answer to RQ2 and RQ4. During the analysis of the participant's answers multiple new ideas possibly worth exploring were mentioned. These can be summarized as follows:

- Creating a coherent *quest line* with communicated sub goals to provide a diverse set of actions and mechanics to follow¹⁰²
- Make the participants solve problems, where the solution has to be found by the participants¹⁰³
- Set up a competition between project members to create additional motivation to find certain elements¹⁰⁴
- Explore a story with command line operations¹⁰⁵
- Place the companion directly within the working context, e.g. within an IDE¹⁰⁶
- Adding more common game mechanics, like fighting enemies (e.g. software bugs)¹⁰⁷
- Visualizing the project history¹⁰⁸
- Creating a character that follows the player within the game world¹⁰⁹
- Implement constraints to the explorable area, so that the player can explore these after each other while gaining knowledge in the process¹¹⁰

These ideas on playful elements provided by the participants complement what was found in literature already and present further options to explore and implement in the future.

Ultimately, an extensive list of possible design guidelines and playful themes has been created through carefully examining research in play and on software development onboarding. This list has been developed through following different perspectives onto play throughout the history on research into it. This was enriched with valuable input (shown above) from software developers generated through a study. Overall, this non-exhaustive list covers the

¹⁰²Coding Units Nr. **26, 27, 33, 36** in *Appendix J*

¹⁰³Coding Unit Nr. **32** in *Appendix J*

¹⁰⁴Coding Unit Nr. **63** in *Appendix J*

¹⁰⁵Coding Unit Nr. **28** in *Appendix J*

¹⁰⁶Coding Unit Nr. **81** in *Appendix J*

¹⁰⁷Coding Units Nr. **103, 105** in *Appendix J*

¹⁰⁸Coding Unit Nr. **106** in *Appendix J*

¹⁰⁹Coding Unit Nr. **112** in *Appendix J*

¹¹⁰Coding Unit Nr. **113** in *Appendix J*

question of *which* playful elements could be used in an onboarding context. Concerning the how of this main research question in regards to the playful elements, this was answered through implementing a subset of this list and discussed through feedback by the study participants. Lastly, the question of how software developers experience such elements was answered through the study data as well. After presenting and summarizing those results within this chapter it is now crucial to critically reflect on the approach, how these relate to other findings in the area and how relevant these findings are. All of this is subject of discussion within the following chapter.

6 Discussion

From guidelines

Discussion of results (justification phase): The critical discussion of the technical or analytical work steps is presented here and the relevance to innovation is established.

Comparisons with other results on similar questions in the literature are entirely appropriate (reference to questions and knowledge gaps in existing literature on the topic identified in the “Related Work” chapter). Arguments may also be presented here concerning social relevance and responsibility with regard to the research question, or it may be placed it in the context of an overall social discourse.

Note for feedback **This is going to be in this part:**

In this part all of the findings from the reuslt chapter, as well as how the research was done, is going to be reflected upon.

6.1 Reflection on the Approach

Note for feedback **This is going to be in this part:**

Reflecting on how well the study approach worked and how valuable the resulting data was. Questions to answer:

- Were the qualitative interviews a good choice?
- How diverse was the interviewee selection?
- How well did the interviews themselves work?
- How valuable was the data and how well could themes be reflected from it?
- What could have been another or better option?

6.2 Related Work

Note for feedback **This is going to be in this part:**

Comparing the results to related work similar to this thesis.

6.3 Relevancy & Future Implications

Possible other implementations:

List here everything that was not done in the probe, but that came up as an idea and put it out there for future research ,e.g.

- Tell a story through git commits (e.g. text-adventure style as an additional medium) -
- Procedurally generated game world, where players can destroy parts of a project to see what breaks due to that
- > And every idea from the participants of the probes

Note for feedback **This is going to be in this part:**

How relevant could all of the research be? Can we start to use it immediately? If not, what has to be done? How could we measure effectiveness in the future?

7 Conclusion

7.1 Contribution

7.2 Outlook

As per the guideline: Include reflection and outlook

The thesis is rounded off with a recapitulation of the main findings and a look ahead to possible further development of the techniques and/or methods chosen. Special attention is paid here to the possibility of innovation in a specific field of application or the further scientific questions that arise from the work.

Note for feedback **This is going to be in this part:**

Concluding the research with a summary of the results and how the questions imposed in this thesis were answered and what that could mean.

List of Figures

1	<i>Vim Adventures</i> Game Scene	31
2	<i>CodinGame</i> Game Scene	32
3	‘Map of design research – research types’ by Liz Sanders [71, p. 3]	45
4	Upper section: ‘Map of design research – new tools and methods’ by Liz Sanders [71, p. 6]	45
5	Adapted IDEO Brainstorming Process (originally by IDEO U [90])	57
6	Probe aesthetics inspiration (including <i>Circles</i> [95] by Jeroen Wimmers (1), <i>Thomas Was Alone</i> [96] by Mike Bithell (2), and <i>Carthasis</i> [97] by Oleg Frolov (3))	63
7	Game objects used throughout the technology probe	64
8	Probe detail scenes with discovered object (1) and object ready for interaction (2)	67
9	Helper companion within the technology probe	68

List of Tables

1	Gamification mechanics and their psychologic effects (condensed by me, based on Sailer et al. [37, p. 373-374])	24
2	Data needed to generate the technology probe game world	69
3	Content Analysis – Category System developed from the research questions .	79

List of Abbreviations

API Application Programming Interface.

CSS Cascading Stylesheets.

CTF Capture The Flag.

HCI Human-Computer Interaction.

HTML Hypertext Markup Language.

IDE Integrated Development Environment.

LAN Local Area Network.

NPM Node Package Manager.

RtD Research through Design.

URL Uniform Resource Locator.

Bibliography

- [1] EVANS DATA CORPORATION (EDC). Global developer population and demographic study, 2019. Vol. 1.
- [2] KULA, R. G., OUNI, A., GERMAN, D. M., AND INOUE, K. On the impact of micro-packages: an empirical study of the npm javascript ecosystem. *arXiv preprint arXiv:1709.04638* (2017).
- [3] SICART, M. *Play matters*. mit Press, 2014.
- [4] GRAZIOTIN, D., FAGERHOLM, F., WANG, X., AND ABRAHAMSSON, P. On the unhappiness of software developers. In *Proceedings of the 21st international conference on evaluation and assessment in software engineering* (2017), pp. 324–333.
- [5] GRAZIOTIN, D., FAGERHOLM, F., WANG, X., AND ABRAHAMSSON, P. What happens when software developers are (un) happy. *Journal of Systems and Software* 140 (2018), 32–47.
- [6] ORTU, M., MURGIA, A., DESTEFANIS, G., TOURANI, P., TONELLI, R., MARCHESI, M., AND ADAMS, B. The emotional side of software developers in jira. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)* (2016), IEEE, pp. 480–483.
- [7] WOBBOCK, J. O. Research contribution types in human-computer interaction. *The Information School University of Washington* (2016).
- [8] FENSON, L., AND SCHELL, R. E. The origins of exploratory play. *Early Child Development and Care* 19, 1-2 (1985), 3–24.
- [9] BALDWIN, D. A., MARKMAN, E. M., AND MELARTIN, R. L. Infants’ ability to draw inferences about nonobvious object properties: Evidence from exploratory play. *Child development* 64, 3 (1993), 711–728.
- [10] HUIZINGA, J. *Homo ludens*. Editora Perspectiva SA, 2020.
- [11] SALEN, K., TEKINBAŞ, K. S., AND ZIMMERMAN, E. *Rules of play: Game design fundamentals*. MIT press, 2004.
- [12] CONSALVO, M. There is no magic circle. *Games and culture* 4, 4 (2009), 408–417.
- [13] COPIER, M. *Beyond the magic circle: A network perspective on role-play in online games*. Utrecht University, 2007.
- [14] RODRIGUEZ, H. The playful and the serious: An approximation to huizinga’s homo ludens. *Game Studies* 6, 1 (2006), 1604–7982.
- [15] KAPROW, A. Essays on the blurring of art and life, ed. J. Kelley. Berkeley: University of California Press (1993).
- [16] ANCHOR, R. History and play: Johan huizinga and his critics. *History and theory* 17, 1 (1978), 63–93.
- [17] BOGOST, I. *Persuasive games*, vol. 5. Cambridge, MA: MIT Press, 2007.
- [18] SUITS, B. *The grasshopper*. University of Toronto Press, 2020.
- [19] SUTTON-SMITH, B. *The ambiguity of play*. Harvard University Press, 2009.

- [20] BERLIN, K. The methodology of socratic dialogue: Creating socratic questions and the importance of being specific. 2004). *Ethics and Socratic Dialogue in Civil Society*. Münster: Lit. Verlag (2004), 148–168.
- [21] MITCHELL, L. Reconsidering the grasshopper: On the reception of bernard suits in game studies. *Game Stud.* 20, 3 (2020).
- [22] BÄCK, A. The paper world of bernard suits. *Journal of the Philosophy of Sport* 35, 2 (2008), 156–174.
- [23] WEIN, E. The journal of american folklore, vol. 113, no. 448. *The Journal of American Folklore* 113, 448 (2000), 213–214.
- [24] HJORTH, L., AND HINTON, S. *Understanding social media*. Sage, 2019.
- [25] PHILLIPS, W., AND MILNER, R. M. *The ambivalent Internet: Mischief, oddity, and antagonism online*. John Wiley & Sons, 2018.
- [26] DANIEL, M., AND GARRY, C. *Video games as culture: considering the role and importance of video games in contemporary society*. Routledge, 2018.
- [27] SPRACKLEN, K. *Digital leisure, the internet and popular culture: Communities and identities in a digital age*. Springer, 2015.
- [28] DETERDING, S., SICART, M., NACKE, L., O’HARA, K., AND DIXON, D. Gamification. using game-design elements in non-gaming contexts. In *CHI ’11 Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2011), CHI EA ’11, Association for Computing Machinery, p. 2425–2428. DOI: <https://doi.org/10.1145/1979742.1979575>.
- [29] SAILER, M., HENSE, J., MANDL, J., AND KLEVERS, M. Psychological perspectives on motivation through gamification. *Interaction Design and Architecture Journal*, 19 (2014), 28–37.
- [30] ANTIN, J., AND CHURCHILL, E. F. Badges in social media: A social psychological perspective. In *CHI 2011 Gamification Workshop Proceedings* (2011), vol. 7, Vancouver, BC.
- [31] WERBACH, K., AND HUNTER, D. *For the win: How game thinking can revolutionize your business*. Wharton digital press, 2012.
- [32] LANDERS, R. N., AND LANDERS, A. K. An empirical test of the theory of gamified learning: The effect of leaderboards on time-on-task and academic performance. *Simulation & Gaming* 45, 6 (2014), 769–785. DOI: <https://doi.org/10.1177/2F1046878114563662>.
- [33] DWECK, C. S. Motivational processes affecting learning. *American psychologist* 41, 10 (1986), 1040.
- [34] NICHOLLS, J. G. Achievement motivation: conceptions of ability, subjective experience, task choice, and performance. *Psychological review* 91, 3 (1984), 328.
- [35] NICHOLSON, S. A recipe for meaningful gamification. In *Gamification in education and business*. Springer, 2015, pp. 1–20. DOI: https://doi.org/10.1007/978-3-319-10208-5_1.
- [36] KAPP, K. M. *The gamification of learning and instruction: game-based methods and strategies for training and education*. John Wiley & Sons, 2012.

- [37] SAILER, M., HENSE, J. U., MAYR, S. K., AND MANDL, H. How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction. *Computers in Human Behavior* 69 (2017), 371–380. DOI: <https://doi.org/10.1016/j.chb.2016.12.033>.
- [38] ZICHERMANN, G., AND LINDER, J. *Game-based marketing: inspire customer loyalty through rewards, challenges, and contests*. John Wiley & Sons, 2010.
- [39] ZICHERMANN, G., AND CUNNINGHAM, C. *Gamification by design: Implementing game mechanics in web and mobile apps*. "O'Reilly Media, Inc.", 2011.
- [40] BOGOST, I. Why gamification is bullshit. *The gameful world: Approaches, issues, applications* (2014), 65–79.
- [41] KNAIVING, K., AND BJÖRK, S. Designing for fun and play: exploring possibilities in design for gamification. In *Proceedings of the first International conference on gameful design, research, and applications* (2013), pp. 131–134. DOI: <https://doi.org/10.1145/2583008.2583032>.
- [42] YATES, R. Y. Onboarding in software engineering.
- [43] BAUER, T. N., AND ERDOGAN, B. Organizational socialization: The effective onboarding of new employees. DOI: <https://doi.org/10.1037/12171-002>.
- [44] CYBENKO, G., AND BREWINGTON, B. The foundations of information push and pull. In *The mathematics of information coding, extraction and distribution*. Springer, 1999, pp. 9–30.
- [45] DAGENAIS, B., OSSHER, H., BELLAMY, R. K., ROBILLARD, M. P., AND DE VRIES, J. P. Moving into a new software project landscape. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1* (2010), pp. 275–284. DOI: <https://doi.org/10.1145/1806799.1806842>.
- [46] BERLIN, L. M., AND JEFFRIES, R. Consultants and apprentices: observations about learning and collaborative problem solving. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work* (1992), pp. 130–137.
- [47] NEVILLE-NEIL, G. V. Code spelunking redux. *Communications of the ACM* 51, 10 (2008), 36–41. DOI: <http://doi.acm.org/10.1145/1400181.1400194>.
- [48] CHERUBINI, M., VENOLIA, G., DELINE, R., AND KO, A. J. Let's go to the whiteboard: how and why software developers use drawings. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2007), pp. 557–566. DOI: <https://doi.org/10.1145/1240624.1240714>.
- [49] BERLIN, L. M. Beyond program understanding: A look at programming expertise in industry. *ESP* 93, 744 (1993), 6–25.
- [50] SIM, S. E., AND HOLT, R. C. The ramp-up problem in software projects: A case study of how software immigrants naturalize. In *Proceedings of the 20th international conference on Software engineering* (1998), IEEE, pp. 361–370.
- [51] ZHOU, M., AND MOCKUS, A. Developer fluency: Achieving true mastery in software projects. In *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering* (2010), pp. 137–146. DOI: <https://doi.org/10.1145/1882291.1882313>.

- [52] DEMARCO, T., AND LISTER, T. *Peopleware: productive projects and teams*. Addison-Wesley, 2013.
- [53] DOMINIC, J., RITTER, C., AND RODEGHERO, P. Onboarding bot for newcomers to software engineering. In *Proceedings of the International Conference on Software and System Processes* (2020), pp. 91–94. DOI: <https://doi.org/10.1145/3379177.3388901>.
- [54] STEINMACHER, I., TREUDE, C., AND GEROSA, M. A. Let me in: Guidelines for the successful onboarding of newcomers to open source projects. *IEEE Software* 36, 4 (2018), 41–49. DOI: <https://doi.org/10.1109/MS.2018.110162131>.
- [55] BOGOST, I. *How to do things with videogames*. U of Minnesota Press, 2011.
- [56] ROBBINS, A., HANNAH, E., AND LAMB, L. *Learning the vi and Vim Editors*. " O'Reilly Media, Inc.", 2008.
- [57] PIERCE, M. C., AND WARE, R. K. Vimtutor. <http://www2.geog.ucl.ac.uk/~plewis/teaching/unix/vimtutor>, Year: *unknown*. Accessed on: 14-08-2021.
- [58] LÓPEZ-FERNÁNDEZ, D., GORDILLO, A., ORTEGA, F., YAGÜE, A., AND TOVAR, E. Lego® serious play in software engineering education. *IEEE Access* 9 (2021), 103120–103131. DOI: <https://doi.org/10.1109/ACCESS.2021.3095552>.
- [59] CHILDERS, N., BOE, B., CAVALLARO, L., CAVEDON, L., COVA, M., EGELE, M., AND VIGNA, G. Organizing large scale hacking competitions. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (2010), Springer, pp. 132–152. DOI: https://doi.org/10.1007/978-3-642-14215-4_8.
- [60] BRISCOE, G. Digital innovation: The hackathon phenomenon.
- [61] KULTIMA, A. Defining game jam. In *FDG* (2015).
- [62] HJORTH, L., PINK, S., HORST, H., SINANAN, J., OHASHI, K., KATO, F., AND ZHOU, B. Beyond emoji play: paralinguistics and intergenerational care-at-distance. *Transnational Migrations in the Asia-Pacific: Transformative Experiences in the Age of Digital Media* (2018), 129–151.
- [63] LATOZA, T. D., VENOLIA, G., AND DELINE, R. Maintaining mental models: a study of developer work habits. In *Proceedings of the 28th international conference on Software engineering* (2006), pp. 492–501.
- [64] STOLTERMAN, E. The nature of design practice and implications for interaction design research. *International Journal of Design* 2, 1 (2008).
- [65] SIMON, H. A. *The sciences of the artificial*. MIT press, 1969.
- [66] COUGHLAN, P., SURI, J. F., AND CANALES, K. Prototypes as (design) tools for behavioral and organizational change: A design-based approach to help organizations change work behaviors. *The journal of applied behavioral science* 43, 1 (2007), 122–134.
- [67] FRAYLING, C. Research in art and design (royal college of art research papers, vol 1, no 1, 1993/4).

- [68] ZIMMERMAN, J., FORLIZZI, J., AND EVENSON, S. Research through design as a method for interaction design research in hci. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2007), pp. 493–502. DOI: <https://doi.org/10.1145/1240624.1240704>.
- [69] ZIMMERMAN, J., AND FORLIZZI, J. Research through design in hci. In *Ways of Knowing in HCI*. Springer, 2014, pp. 167–189. DOI: https://doi.org/10.1007/978-1-4939-0378-8_8.
- [70] ZIMMERMAN, J., STOLTERMAN, E., AND FORLIZZI, J. An analysis and critique of research through design: towards a formalization of a research approach. In *proceedings of the 8th ACM conference on designing interactive systems* (2010), pp. 310–319.
- [71] SANDERS, L. An evolving map of design practice and design research. *interactions* 15, 6 (2008), 13–17.
- [72] AKAMA, Y., AND IVANKA, T. What community? facilitating awareness of ‘community’ through playful triggers. In *Proceedings of the 11th biennial participatory design conference* (2010), pp. 11–20. DOI: <https://doi.org/10.1145/1900441.1900444>.
- [73] GAVER, B., DUNNE, T., AND PACENTI, E. Design: cultural probes. *interactions* 6, 1 (1999), 21–29.
- [74] CRABTREE, A., HEMMINGS, T., RODDEN, T., CHEVERST, K., CLARKE, K., DEWSBURY, G., HUGHES, J., AND ROUNCEFIELD, M. Designing with care: Adapting cultural probes to inform design in sensitive settings. In *Proceedings of the 2004 Australasian Conference on Computer-Human Interaction (OZCHI2004)* (2003), pp. 4–13.
- [75] WYETH, P., AND DIERCKE, C. Designing cultural probes for children. In *Proceedings of the 18th Australia conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments* (2006), pp. 385–388. DOI: <https://doi.org/10.1145/1228175.1228252>.
- [76] KJELDSKOV, J., GIBBS, M., VETERE, F., HOWARD, S., PEDELL, S., MECOLES, K., AND BUNYAN, M. Using cultural probes to explore mediated intimacy. In *Using Cultural Probes to Explore Mediated Intimacy* (2004), CHISIG.
- [77] CELIKOGLU, O. M., OGUT, S. T., AND KRIPPENDORFF, K. How do user stories inspire design? a study of cultural probes. *Design Issues* 33, 2 (2017), 84–98. DOI: https://doi.org/10.1162/DESI_a_00441.
- [78] FISHER, K. E., YEFIMOVA, K., AND BISHOP, A. P. Adapting design thinking and cultural probes to the experiences of immigrant youth: Uncovering the roles of visual media and music in ict wayfaring. In *Proceedings of the 2016 CHI conference extended abstracts on human factors in computing systems* (2016), pp. 859–871. DOI: <https://doi.org/10.1145/2851581.2851603>.
- [79] HUTCHINSON, H., MACKAY, W., WESTERLUND, B., BEDERSON, B. B., DRUIN, A., PLAISANT, C., BEAUDOUIN-LAFON, M., CONVERSY, S., EVANS, H., HANSEN, H., ET AL. Technology probes: inspiring design for and with families. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2003), pp. 17–24. DOI: <https://doi.org/10.1145/642611.642616>.

- [80] IVERSEN, O. S., AND NIELSEN, C. Using digital cultural probes in design with children. In *Interaction Design And Children: Proceeding of the 2003 conference on Interaction design and children* (2003), vol. 1, pp. 154–154.
- [81] MATTELMÄKI, T., ET AL. *Design probes*. Aalto University, 2006.
- [82] FITTON, D., CHEVERST, K., ROUNCEFIELD, M., DIX, A., AND CRABTREE, A. Probing technology with technology probes. In *Equator Workshop on Record and Replay Technologies* (2004), Citeseer.
- [83] O'BRIEN, S., AND MUELLER, F. Holding hands over a distance: technology probes in an intimate, mobile context. In *Proceedings of the 18th Australia Conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments* (2006), pp. 293–296. DOI: <https://doi.org/10.1145/1228175.1228226>.
- [84] EDWARDS, H. M., McDONALD, S., AND ZHAO, T. Exploring teenagers' motivation to exercise through technology probes. In *Proceedings of HCI 2011 The 25th BCS Conference on Human Computer Interaction 25* (2011), pp. 104–113. DOI: <https://doi.org/10.14236/ewic/HCI2011.34>.
- [85] FLICK, U. *An introduction to qualitative research*. sage, 2018.
- [86] WITZEL, A., AND REITER, H. *The problem-centred interview*. Sage, 2012.
- [87] BRAUN, V., AND CLARKE, V. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.
- [88] SCHREIER, M. *Qualitative content analysis in practice*. Sage publications, 2012.
- [89] SCHREIER, M. Ways of doing qualitative content analysis: disentangling terms and terminologies. In *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research* (2014), vol. 15. DOI: <https://doi.org/10.17169/fqs-15.1.2043>.
- [90] IDEO U. Brainstorming. <https://www.ideo.com/pages/brainstorming>, 2021. accessed on 19th of August 2021.
- [91] FRIIS DAM, R., AND SIANG, T. Y. Learn how to use the best ideation methods: Brainstorming, braindumping, brainwriting, and brainwalking. <https://www.interaction-design.org/literature/article/learn-how-to-use-the-best-ideation-methods-brainstorming-braindumping-brainwriting-and-brainwalking>, 2020. accessed on 18th of August 2021.
- [92] STACKOVERFLOW INSIGHTS. 2021 developer survey. <https://www.interaction-design.org/literature/article/learn-how-to-use-the-best-ideation-methods-brainstorming-braindumping-brainwriting-and-brainwalking>, 2021. accessed on 11th of August 2021.
- [93] JASPAN, C., JORDE, M., KNIGHT, A., SADOWSKI, C., SMITH, E. K., WINTER, C., AND MURPHY-HILL, E. Advantages and disadvantages of a monolithic repository: a case study at google. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice* (2018), pp. 225–234. DOI: <https://doi.org/10.1145/3183519.3183550>.
- [94] NEALEN, A., SALTSMAN, A., AND BOXERMAN, E. Towards minimalist game design. In *Proceedings of the 6th International Conference on Foundations of Digital Games* (2011), pp. 38–45. DOI: <https://doi.org/10.1145/2159365.2159371>.

- [95] WIMMERS, J. Circles. https://store.steampowered.com/app/460250/_Circles/, 2017. accessed on 12th of August 2021.
- [96] WIMMERS, J. Circles. https://store.steampowered.com/app/460250/_Circles/, 2017. accessed on 12th of August 2021.
- [97] BITHELL, M. Thomas was alone. <http://www.mikebithellgames.com/thomaswasalone/>, Year Unknown. accessed on 12th of August 2021.
- [98] STACKOVERFLOW. At what point/range is a code file too big? <https://softwareengineering.stackexchange.com/questions/176999/at-what-point-range-is-a-code-file-too-big>, 2012. accessed on 12th of August 2021.

Appendices

A Data Protection & Consent Form

Information about data protection and declaration of consent

Dear participant,

thank you for your interest in participating in this study on "Playful Experiences in the Onboarding Process of Software Development Projects". This study takes place as a part of my master thesis on "Onboarding in Software Development Projects as a Space of Play". The thesis is written as a part of the new Joint Master "Human-Computer Interaction" of the FH Salzburg and the University of Salzburg. In order to use your personal data, we would need your written declaration of consent. Please, carefully read the following information.

General Study Information

The goal of this study is to research how software developers experience playful approaches on the onboarding process into new projects. The results of this study are going to contribute to the research body of both the software development field but also the field of "Play". From the study results conclusions are going to be drawn about: The openness towards playful experiences in the software development field, possible hurdles on implementing such experiences, the general attitude of developers towards play in the field and the selection of what information to include in onboarding processes.

Overall Study Procedure

The participant will take part in a hybrid-study setting. Part of the study consists of going through an interactive visualization of an open-source software project and revealing parts of information within some parts of the project. The other part of the study consists of questions on the visualization and general questions on the intersection of Play and software development. The study can be accessed online and can be finished online without additional work on the participants side. Overall the procedure should take between 15 and 25 minutes.

Rights

The participation in this study is voluntary. You may withdraw or stop your participation at any time without incurring any consequences. Moreover, you can request information about

the processed personal data at any time, even after the termination of the study (for further information see also Confidentiality and data processing). If you have any questions please contact the study leader.

Risks

You do not incur any risks by participating in this study. If you feel uncomfortable during the study you can cancel at any time without giving reasons and without incurring any consequences.

Confidentiality and Data Processing

Your data will only be processed within the aforementioned project if you give your written consent. The data is going to be processed via the Firebase platform (<https://firebase.google.com/>) on european servers. For additional information on their privacy and data protection guidelines visit their documentation at <https://policies.google.com/u/0/privacy>. The following data is going to be processed within the study:

- Demographic Information (Name, Age, Professional Background, Years of Experience, Gender)
- Your actions within the visualization (Timestamps of Clicks and Mouse moves together with the timestamps of certain events)
- Your answers to the questions that are part of the study

After the end of the study, the data is going to be exported from Firebase and the underlying app instance is going to be deleted. This data is going to be kept strictly confidential and is not going to be passed on to further third parties. The data is going to be analyzed and then used within my master thesis. If you only want to be mentioned anonymously in my thesis please indicate so at the bottom. If you are doing that your name and age are going to be hidden in my thesis and your data is only going to be connected to a random ID.

Remuneration

You will not receive an expense allowance for participation.

Consent

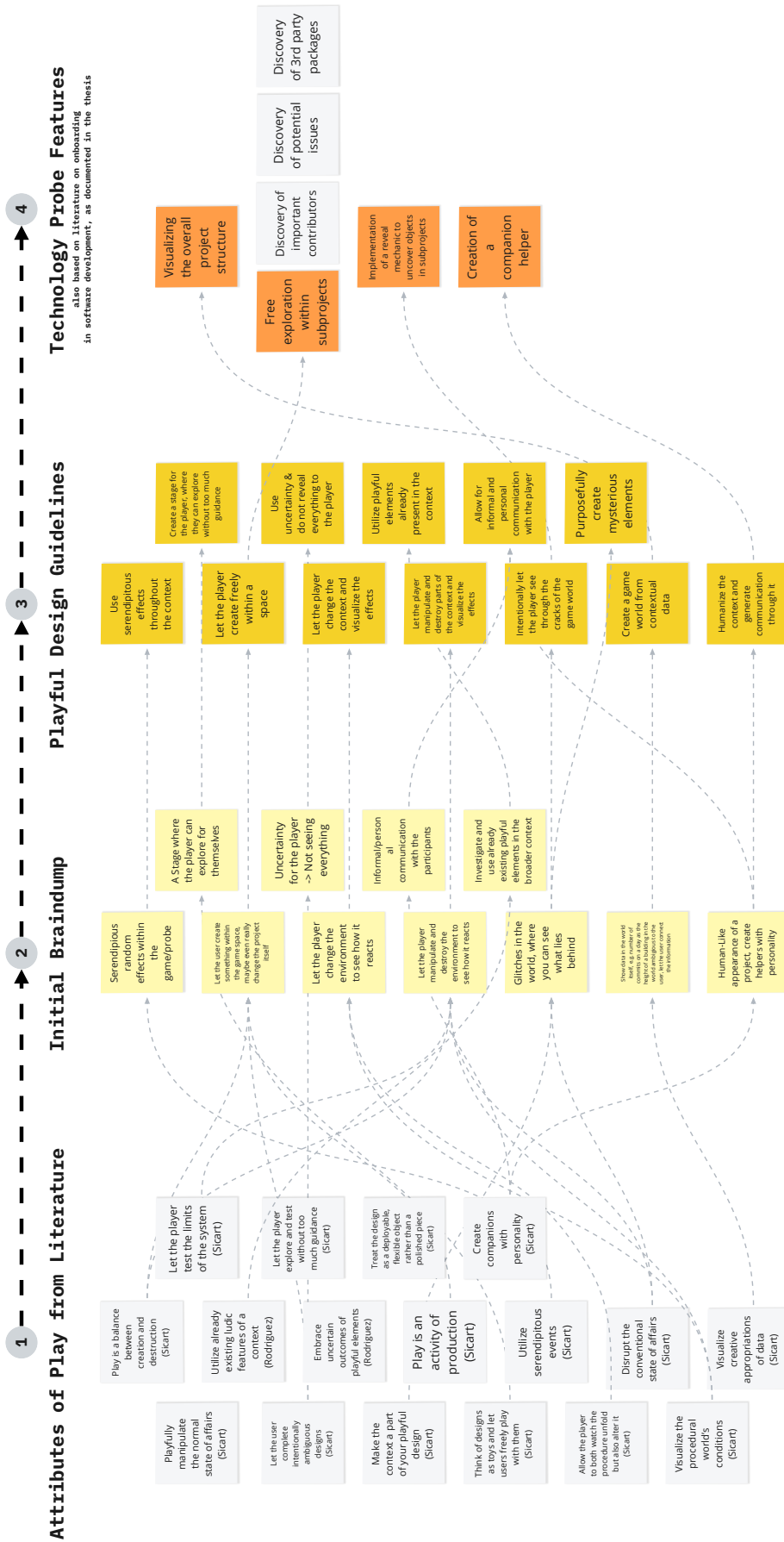
I have carefully read and understood the declaration of consent. Any additional questions were answered to my satisfaction. My participation is voluntary, and I know that I can withdraw at any time, even without giving reasons, without incurring any disadvantages. By clicking on “Agree” below I agree to the processing of my personal data collected for this study and I am willing to participate.

Anonymity

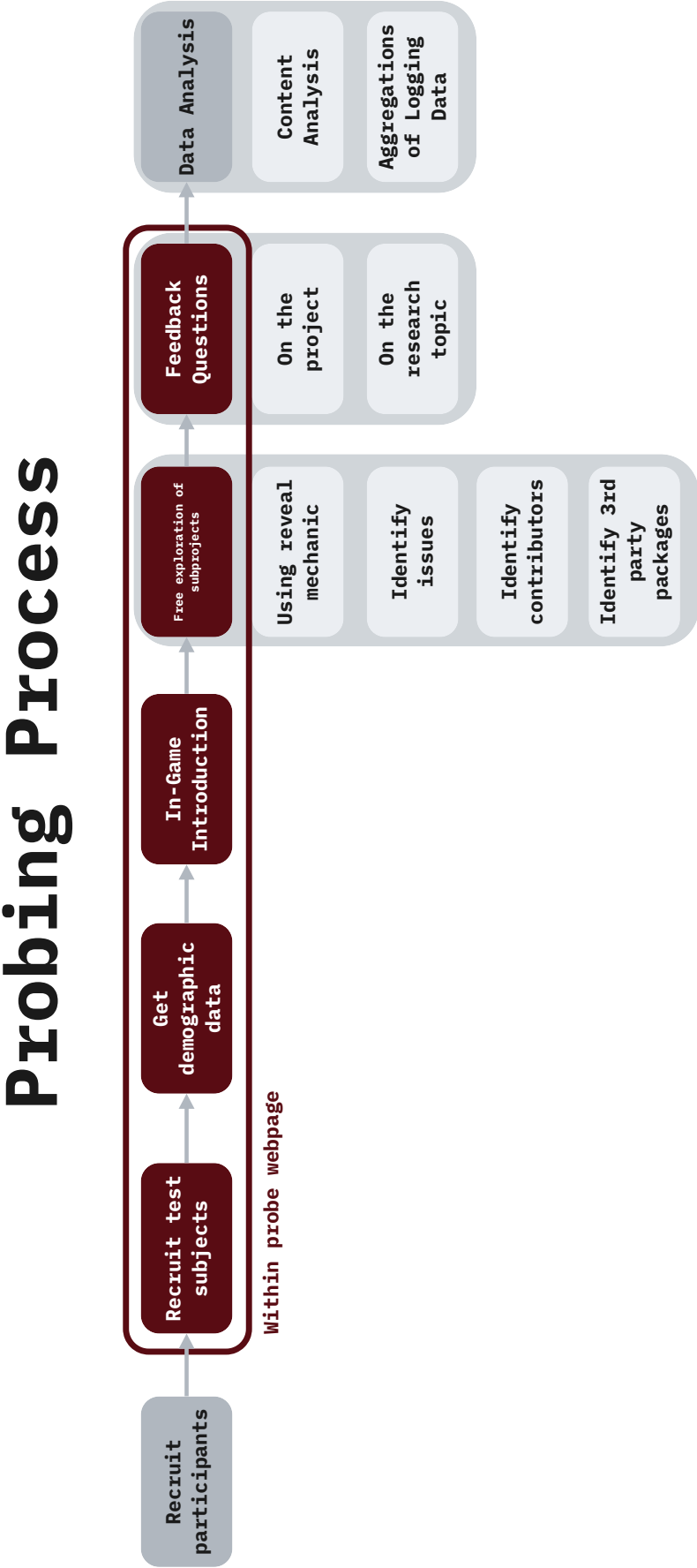
I want to participate anonymously (the data is not going to be connected to your name)

B Probe Development Process

From Theory to Probe



C Probe Execution Process



D Probe Information Message Types

3rd-party package



NPM PACKAGE

merkle-patricia-tree

This package is used throughout this part of the repository. Below you can see the version that is installed currently. If you want to take a look at the package documentation, past versions and much more, click the button in the lower right corner. It'll lead you to the npmjs website of the package.

This package is installed at

^4.2.0

[Check this out in detail](#)

Contributor



CONTRIBUTOR

holgerd77

This is one of the main contributors to the overall repository and this part of the repository specifically. Below you can see the contributors latest commits and the button on the right will take you straight to the respective Github profile.

LAST COMMITS

Client: central event bus implementation (#1187)
July 17, 2021 4:01 AM • [Take a look on Github](#)

Merge pull request #1165 from ethereumjs/add-client-cli-test client: add basic cli test
July 12, 2021 11:51 AM • [Take a look on Github](#)

Merge pull request #1327 from ethereumjs/new-releases New Library Releases (tx.supports() integration, Common custom chains + enum Chain/HF, Util v7.1.0)
July 8, 2021 11:08 AM • [Take a look on Github](#)

[Check this out in detail](#)

Potential Issue (Large File)



LEGACY ALERT

block.ts

This file seems pretty big compared to the other files in this part of the project. It's 15740 bytes and has 487 lines of code. That could make it hard to read for new contributors and people looking at the repository. You might want to look into a refactoring in order to keep it at a more readable size. What you could also do is contact one of the contributors to help in understanding what's going on. They must be hiding somewhere in this package as well, try to reveal them to access their Github profiles.

Excerpt from the file:

```
public static fromBlockData(blockData: BlockData = {}, opts?: BlockOptions) {
  const { header: headerData, transactions: txsData, uncleHeaders: uhsData } = blockData
  const header = BlockHeader.fromHeaderData(headerData, opts)

  // parse transactions
  const transactions = []
  for (const txData of txsData ?? []) {
    const tx = TransactionFactory.fromTxData(txData, {
      ...opts,
```

[Check this out in detail](#)

E Probe Data Collection – Questions

Demographic Information

Text Input: **Your Name**

Number Input: **Your Age**

Text Input: **Your Professional Background**

Select Input: **Your Professional Experience** (Options: 0-1 years, 1-3 years, 3-5 years, 5-10years)

Questions on the underlying project

- 1) Do you remember any of the contributors of this project? Please name those that you remember below
- 2) What were the two largest packages of this monorepo (not the npm packages, but the ones the monorepo is made out of)?
- 3) Which of the projects "sub"-packages were used most throughout the other subpackages?
- 4) Any npm packages you've revealed in this project, that you can think of? Did you take a look at their respective websites?
- 5) Have you taken a look at any of the files that were potentially "legacy" files? If yes, did you agree on them being "refactor-worthy" or do you think they were fine as-is?

Questions on the research topic

- 1) Would you say, that you have discovered something interesting about the underlying project from going through this interactive visualization? If yes, what? If no, what was missing or what would you have liked to see?
- 2) What was your overall experience going through this visualization? What did you like or did not like?
- 3) How did you experience the companion (lower right)? Was it helpful, annoying or did you not really interact with it at all?
- 4) Could you imagine yourself using some kind of interactive visualization or something similar on different projects to learn about them? If so, on which projects would you want to try it out? If not, what would you prefer instead to make yourself familiar with new projects?
- 5) How do you usually approach the onboarding onto new projects? To whom do you talk to, what applications do you use, how do you navigate through code?

- 6) Do you work on Open Source Projects? If you are a contributor of one, how do you try to make it easy for new collaborators to work on these projects? If not, how do you approach collaborating for yourself?
- 7) Do you have any additional ideas on how playful elements or game mechanics could be used within the onboarding phase of software development projects?
- 8) What is your general stance on using games/game mechanics or playful elements within software development? Do you see a difference of using such mechanics in open source software vs. in a work environment?
- 9) Do you have any additional ideas on how playful elements or game mechanics could be used within the onboarding phase of software development projects? Any elements from games you play that you think could be reused when making yourself familiar with new projects?
- 10) Anything else you want to mention?

F Probe Data Collection – Logging

Logging Single Event Format

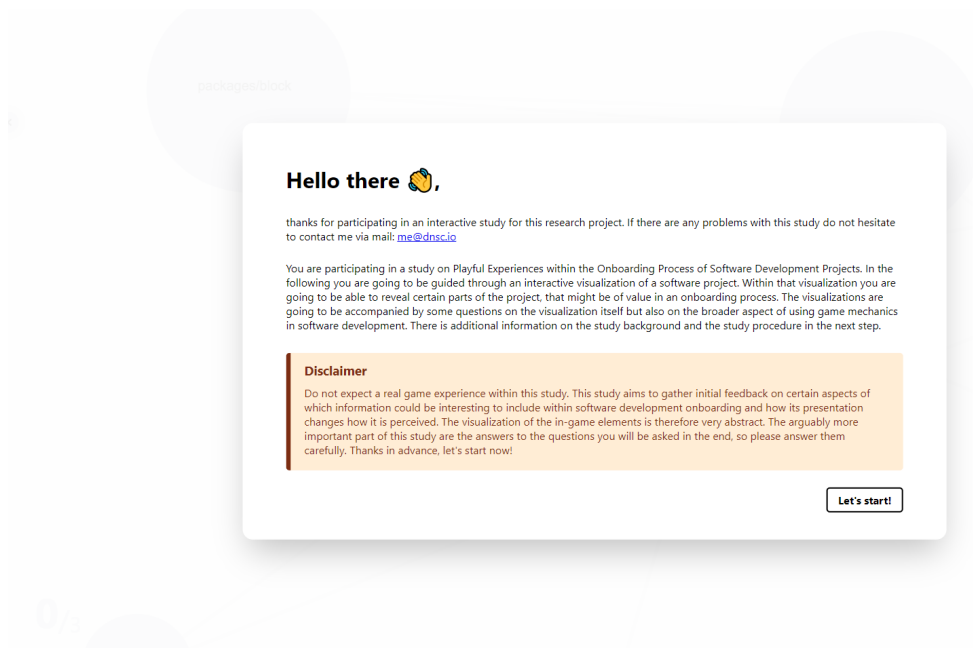
- **message:** Additional information depending on the event type
- **timestamp:** The unix timestamp at which the event occurred
- **type:** The type of interaction according to the event codes below

Logging Event Codes with their respective interactions

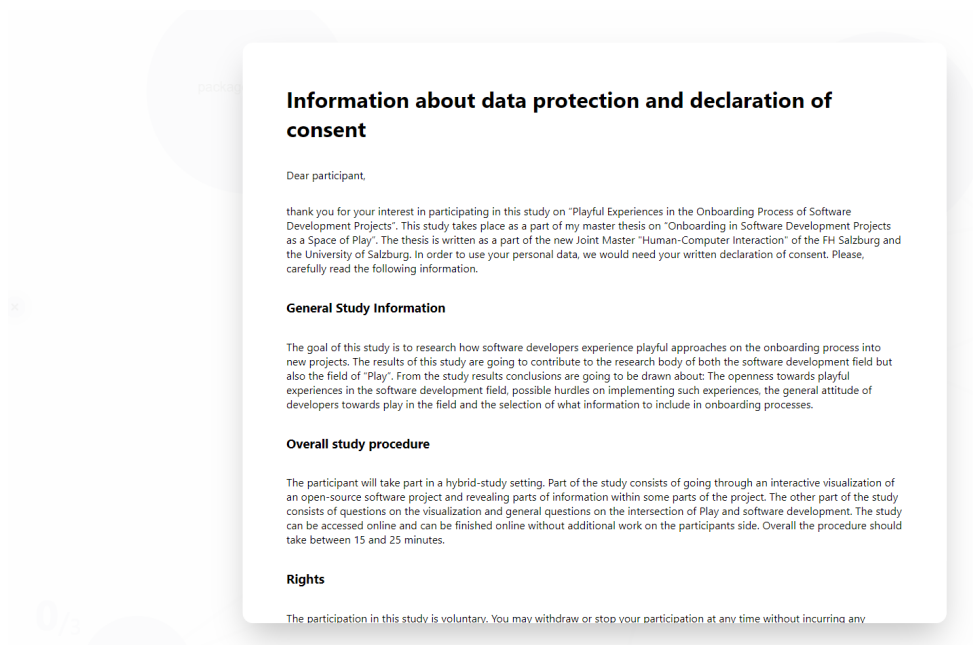
Event Code	Interaction
CC	Click on companion
OC	Overview click
RC	Reveal click
LI	Legacy identified
LR	Legacy revealed
PI	Package identified
PR	Package revealed
NI	Contributor identified
NR	Contributor revealed
SF	Subproject finished
GF	Game finished

G Technology Probe Scenes

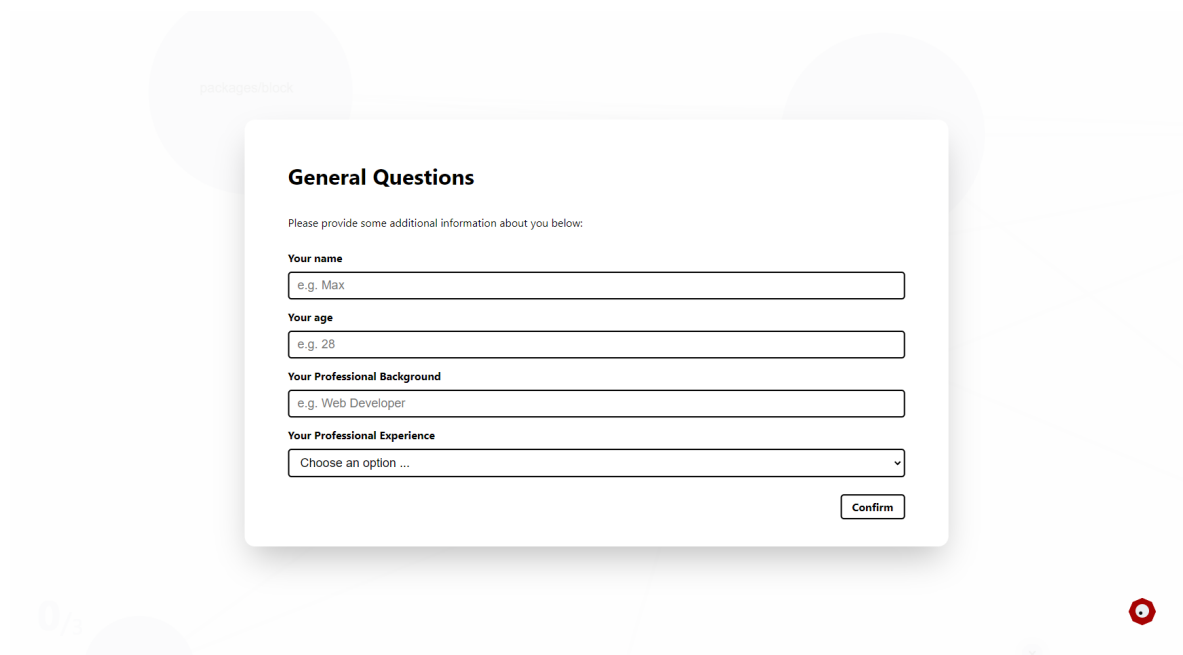
Step 1: Introduction



Step 2: Consent & Data Protection



Step 3: Demographic Questions



The screenshot shows a web application interface with a central modal titled "General Questions". The modal contains a prompt "Please provide some additional information about you below:" followed by four input fields: "Your name" (with placeholder "e.g. Max"), "Your age" (with placeholder "e.g. 28"), "Your Professional Background" (with placeholder "e.g. Web Developer"), and "Your Professional Experience" (a dropdown menu with "Choose an option ..." and a downward arrow). A "Confirm" button is at the bottom right of the modal. The background is a light gray with faint circular patterns and the text "packages/block". A red circular icon with a white dot is visible on the right side of the screen.

General Questions

Please provide some additional information about you below:

Your name
e.g. Max

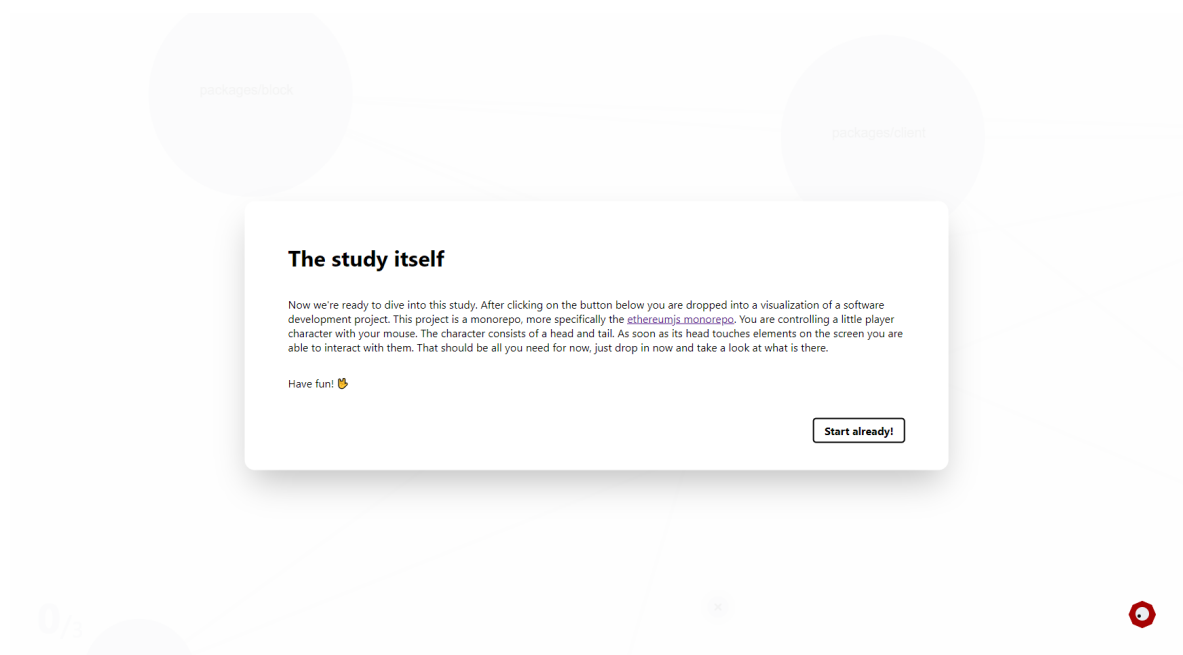
Your age
e.g. 28

Your Professional Background
e.g. Web Developer

Your Professional Experience
Choose an option ...

Confirm

Step 4: Visualization Introduction



The screenshot shows a web application interface with a central modal titled "The study itself". The modal contains a paragraph of text explaining the study: "Now we're ready to dive into this study. After clicking on the button below you are dropped into a visualization of a software development project. This project is a monorepo, more specifically the `ethereumjs/monorepo`. You are controlling a little player character with your mouse. The character consists of a head and tail. As soon as its head touches elements on the screen you are able to interact with them. That should be all you need for now, just drop in now and take a look at what is there." Below the text is the phrase "Have fun! 🥳". A "Start already!" button is at the bottom right of the modal. The background is a light gray with faint circular patterns and the text "packages/block". A red circular icon with a white dot is visible on the right side of the screen.

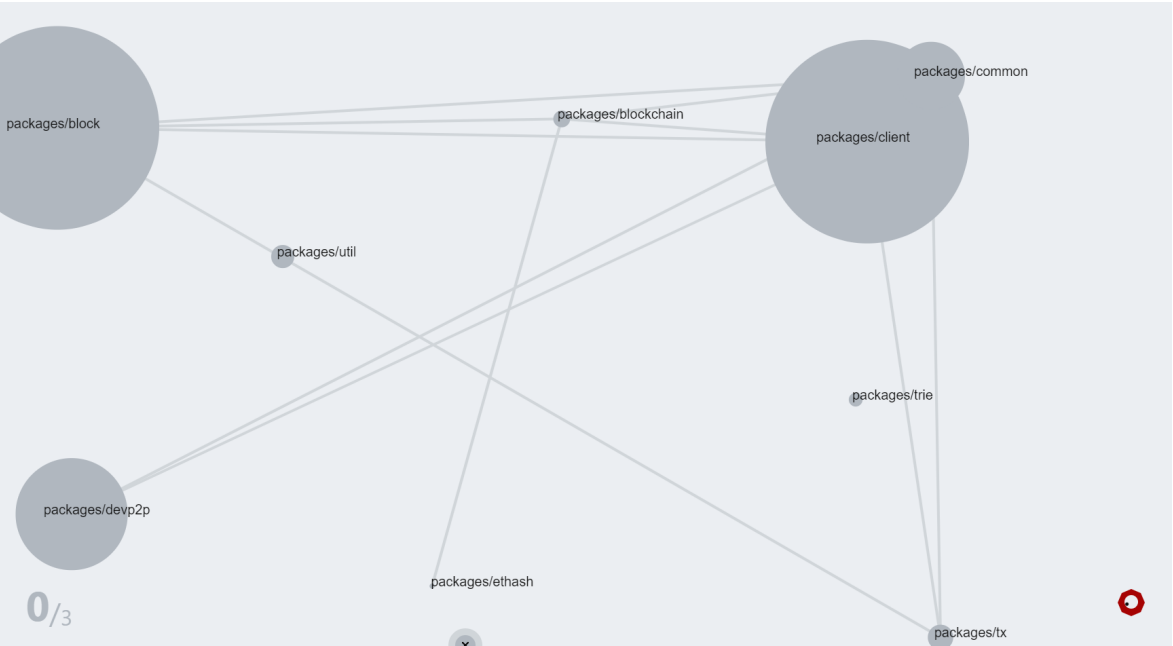
The study itself

Now we're ready to dive into this study. After clicking on the button below you are dropped into a visualization of a software development project. This project is a monorepo, more specifically the `ethereumjs/monorepo`. You are controlling a little player character with your mouse. The character consists of a head and tail. As soon as its head touches elements on the screen you are able to interact with them. That should be all you need for now, just drop in now and take a look at what is there.

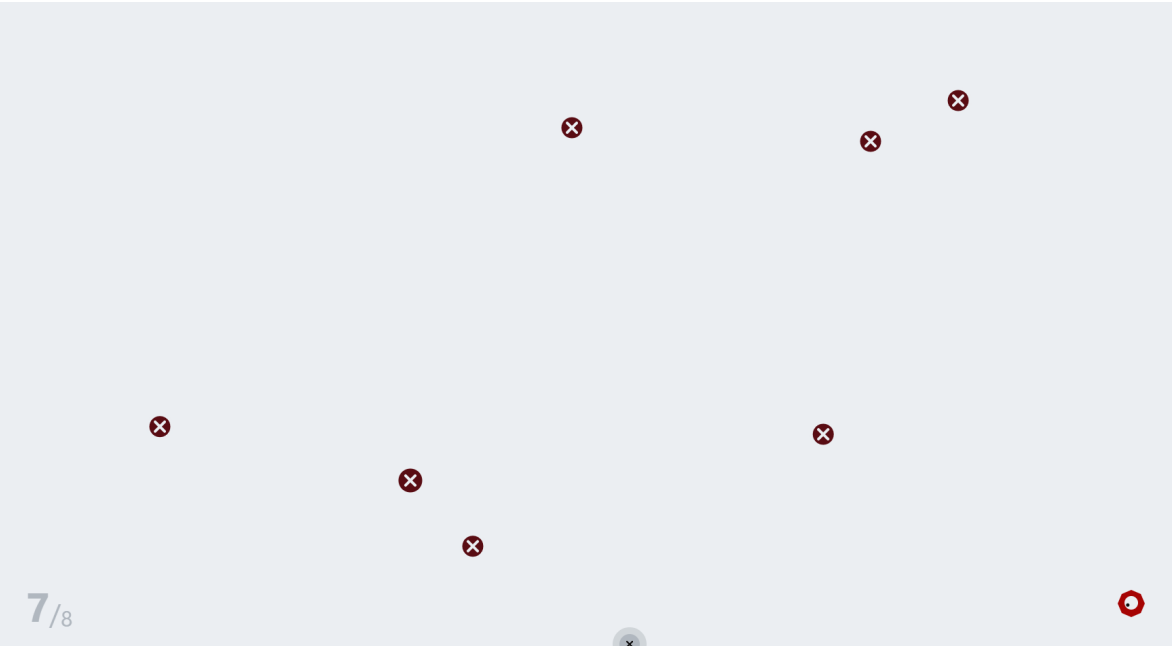
Have fun! 🥳

Start already!

Step 5: Probe Overview Scene (Example)



Step 6: Probe Detail Scene (Example)



Step 7: Questions on the underlying project

packages/block

3/3

Questions about the ethereumjs project

Thanks for going through this visualization. It would be great if you could answer a set of questions now:

1) Do you remember any of the contributors of this project? Please name those that you remember below

2) What were the two largest packages of this monorepo (not the npm packages, but the ones the monorepo is made out of)?

3) Which of the projects "sub"-packages were used most throughout the other subpackages?

4) Any npm packages you've revealed in this project, that you can think of? Did you take a look at their respective websites?

signals/common

packages/tx

Step 8: Questions on the research topic

packages/block

3/3

Questions about Play in Software Development

After the project-specific questions in the last step I want to get your valuable input as a developer for the following questions:

1) Would you say, that you have discovered something interesting about the underlying project from going through this interactive visualization? If yes, what? If no, what was missing or what would you have liked to see?

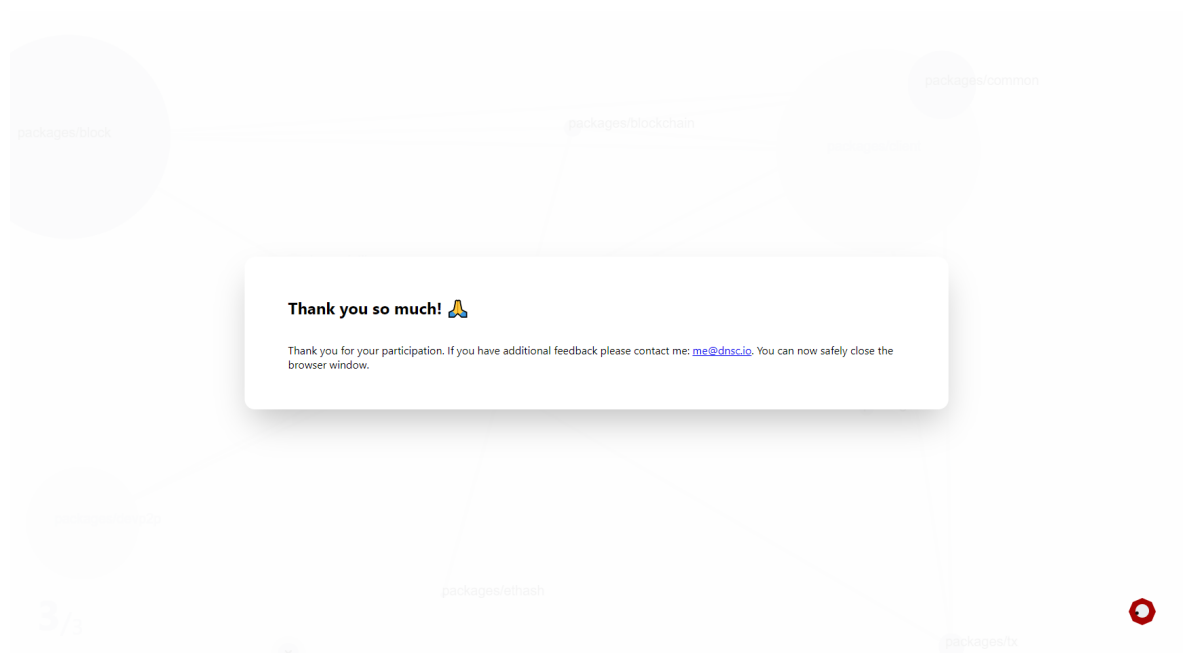
2) What was your overall experience going through this visualization? What did you like or did not like?

3) How did you experience the companion (lower right)? Was it helpful, annoying or did you not really interact with it at all?

signals/common

packages/tx

Step 9: Outro



H Participants

Code	Name	Age	Professional Background	Professional Experience
P1	Anonymous	31	Web Developer	0-1 year
P2	Anonymous	28	Full Stack Developer	1-3 years
P3	Jens	26	Developer	5-10 years
P4	Lukas	28	Business Development	5-10 years
P5	Paolo Sfilio	28	Software Developer	3-5 years
P6	Anonymous	36	Backend Developer	10+ years
P7	Anonymous	28	Data Scientist	1-3 years
P8	Simon Hansen	27	Web Developer	3-5 years
P9	Anonymous	24	.Net Dev	3-5 years

I Aggregated Data from the Technology Probe Logs

Following are the general data points for each participant, aggregated from the raw logs of the technology probe. This text was generated programmatically, the respective script can be found within the accompanying repository at

`./supplementary-material/data-analysis/aggregate-logs/aggregate-logs.js`.

Participants

Participant P1

This participant spent a total time of **5 minutes 19 seconds** within the interactive part of the technology probe.

During that time the participant clicked within the overview and to reveal (in the detail scenes) **119** times. The participant also tried to click the companion **8** times.

Participant P2

This participant spent a total time of **6 minutes 49 seconds** within the interactive part of the technology probe.

During that time the participant clicked within the overview and to reveal (in the detail scenes) **589** times. The participant also tried to click the companion **13** times.

Participant P3

This participant spent a total time of **12 minutes 50 seconds** within the interactive part of the technology probe.

During that time the participant clicked within the overview and to reveal (in the detail scenes) **224** times. The participant also tried to click the companion **10** times.

Participant P4

This participant spent a total time of **3 minutes 15 seconds** within the interactive part of the technology probe.

During that time the participant clicked within the overview and to reveal (in the detail scenes) **131** times. The participant also tried to click the companion **7** times.

Participant P5

This participant spent a total time of **null [ed. note: not finished]** within the interactive part of the technology probe.

During that time the participant clicked within the overview and to reveal (in the detail scenes) **40** times. The participant also tried to click the companion **0** times.

Participant P6

This participant spent a total time of **2 minutes 18 seconds** within the interactive part of the technology probe.

During that time the participant clicked within the overview and to reveal (in the detail scenes) **131** times. The participant also tried to click the companion **4** times.

Participant P7

This participant spent a total time of **8 minutes 14 seconds** within the interactive part of the technology probe.

During that time the participant clicked within the overview and to reveal (in the detail scenes) **561** times. The participant also tried to click the companion **45** times.

Participant P8

This participant spent a total time of **9 minutes 50 seconds** within the interactive part of the technology probe.

During that time the participant clicked within the overview and to reveal (in the detail scenes) **185** times. The participant also tried to click the companion **6** times.

Participant P9

This participant spent a total time of **5 minutes 50 seconds** within the interactive part of the technology probe.

During that time the participant clicked within the overview and to reveal (in the detail scenes) **184** times. The participant also tried to click the companion **6** times.

J Qualitative Data – Coding Units

On the following pages, the documented coding units for the qualitative content analysis can be found. These are the result of a multi-step process as described in chapter 5. The working document that includes every step of the process (Selection, Marking, Categories, Coding) can be accessed at the supplementary repository at the following path:

```
./supplementary-material/data-analysis/qualitative_analysis.xlsx
```

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P1	didn't really care when I did this because I wasn't aware this was a thing	Experience towards (playful) features	Contributor Revealable		1
P1	no emotional attachment to this project	Contextual Considerations			2
P1	Because contributors don't really help me to get started with a project I guess	Social Onboarding			3
P1	only had some 80 lines and I thought to myself 'well, that's not that big in my opinion'	Experience towards (playful) features	Potential Issue Revealable		4
P1	I didn't really get the impression of the existence of 'sub' packages, whatever they are supposed to be anyway	Experience towards (playful) features	Overview Scene (Subpackages)		5
P1	only check out npm packages when I actually have to work with them, not randomly only because they are used in a project	Technical Onboarding			6
P1	Not exactly sure what you mean with 'legacy' file	Experience towards (playful) features	Potential Issue Revealable		7

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P1	actually interesting to get told about contributors and see their faces. Not that it would really help me getting into this project but yeah, gives it a more personal flavor	Social Onboarding			8
P1	during each discovery was still a bit big to digest in a playfull way I think	Attitude towards Playful elements			9
P1	visualizations like images/icons rather than just plain text and a sample of the code and a link to the actual file	Improvements for implemented features			10
P1	or an icon I learned earlier this represents a contributor	Improvements for implemented features			11
P1	The texts where pretty repetitive and I stopped reading after 'contributor' so I might've missed any information that was unique to this slide	Experience towards (playful) features			12

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P1	I was a bit confused because I didn't know what to do when I reached the first blank slide	Experience towards (playful) features		Can also be seen in the logs, No action at all in the beginning, until companion message in both detail and overview scene (see logs lines 10-63)	13
P1	helper on the bottom was quick to assist, though	Experience towards (playful) features			14
P1	elements disappear, I would make them fade out rather than just getting invisible instantly	Improvements for implemented features			15
P1	impression of revealing areas better when the initial background would be dark and when you trigger the sonar, everything inside the expanding circle is bright. Maybe with some something visual in the background that gives some depth perception.	Improvements for implemented features			16

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P1	came to help always at the right time. Good guy!	Experience towards (playful) features		Both companion helping messages were shown, as seen in the logs (log lines 10, 60)	17
P1	To be honest, I don't really know. It strongly depends on the urgency to get familiar with a project I guess	Attitude towards Play in Onboarding			18
P1	Exploring something like in this example can be frustrating if you really just want to get the gist of something because you stumble across so many random things	Attitude towards Play in Onboarding			19
P1	this kind of visualization isn't the right thing to learn about a coding project. At least not for me.	Attitude towards Play in Onboarding			20
P1	hadn't had too much of these occasions yet	Organizational Onboarding			21
P1	I set it up first to get a feeling for what I'm dealing with (if possible) and than I dig deeper into it	Technical Onboarding			22

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P1	I talk to those who set up or work on the project if possible	Social Onboarding			23
P1	for my IDE, it's mostly stuff that helps visualize things like pairs of brackets and key words or variables vs functions and so on.	Technical Onboarding			24
P1	I try to stick to conventions and write code that's hopefully easy to understand with meaningful variable and function names	Technical Onboarding			25
P1	I'd provide some extra goals, like the first 5 questions about contributors I remember or not and so on so the explorer has something to aim for	Ideas on Playful Elements for Onboarding			26
P1	'quests' can be aimed to point out core concepts of the project and guide the participant towards them.	Ideas on Playful Elements for Onboarding			27
P1	Gamification is great to learn things like coding itself, like exploring a story with command line operations	Ideas on Playful Elements for Onboarding			28

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P1	Exploring something in a playful way also needs to be done it each person's personal pace	Attitude towards Play in Onboarding			29
P1	exploring something also needs time and is not fun if you are under pressure and you just want to get the gist of something	Attitude towards Play in Onboarding			30
P1	As you did it in your example, I want to have the headspace to branch out and check out parts of the project that are maybe just nice to know but necessarily helping me with actually working on the project	Attitude towards Play in Onboarding			31
P1	make the participants make solving problems, like searching for an answer for a specific question	Ideas on Playful Elements for Onboarding			32
P1	let the participant choose a goal and give him or her some sort of quest line with a bunch of sub goals to achieve	Ideas on Playful Elements for Onboarding			33

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P1	letting them to search for reusable components which are used throughout the project to make them aware they exist	Ideas on Onboarding Elements			34
P1	what dependencies a main component has. Then you have just a portion of npm packages to digest	Ideas on Onboarding Elements			35
P1	Serve the project guided and bit by bit, only giving participants the choice what main goal they have or they want to achieve by playing this 'quest line'?	Ideas on Playful Elements for Onboarding			36
P3	Ich habe mir aber ein paar Profile angeguckt	Experience towards (playful) features	Contributor Revealable		37
P3	Anhang der Größe und Zeilenmenge hätte ich dem \"refactor-worthy\" aber in den meisten Fällen vermutlich zugestimmt.	Experience towards (playful) features	Potential Issue Revealable		38
P3	ungefähre Vorstellung davon bekommen, wie die Projekte untereinander zusammenhängen	Experience towards (playful) features			39

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P3	wenn ich es mir nicht merken konnte, war eine Visualisierung auf alle Fälle hilfreich	Experience towards features (playful)			40
P3	schön durch die Vorstellung der Contributors ein paar Namen kennenzulernen, denn man im weiteren Verlauf des Projekt vermutlich häufiger über den Weg laufen wird	Experience towards features (playful)	Contributor Revealeable		41
P3	Mit der Vorstellung der großen \"refactor-worthy\" Dateien werden einem direkt mehrere Möglichkeiten eröffnet, in das Projekt einzusteigen	Experience towards features (playful)	Potential Issue Revealeable		42
P3	Mir fehlte ein wenig das fachliche über das Projekt. Was tut es? Wofür ist es gut? Nur anhand des Codes ist das schwierig zu verstehen.	Improvements for implemented features			43
P3	Im ersten Moment wusste ich nicht, wie ich starten sollte	Experience towards features (playful)		As seen in the logs, companion messages appeared in both scenes (lines 9040, 9090)	44

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P3	urch die Hilfen habe ich aber sehr schnell verstanden, wie der Ablauf gedacht ist	Experience to-wards (playful) features		Both messages seen, after that clicking started as can be seen in logs (9040 and following, 9090 and following)	45
P3	Suchen der verschiedenen Punkte hat im ersten Moment Spaß gemacht, wurde anschließend aber ein bisschen nervig (vor allem weil einige Punkte vor einem abgehauen sind).	Experience to-wards (playful) features			46
P3	hilfreich um zu verstehen, wie das Onboarding zu bedienen ist	Experience to-wards (playful) features			47
P3	Leider sagt er nichts, wenn man auf ihn klick	Experience to-wards (playful) features			48
P3	Wiederholung der Hilfe oder irgendein Feedback wären nett gewesen.	Improvements for implemented features			49

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P3	ganz groben ersten Überblick über das Projekt zu verschaffen, würde ich so eine Visualisierung durchaus gut finden	Attitude towards Play in Onboarding			50
P3	Wenn ich an einem neuen Projekt anfangen, arbeite ich mich von außen immer weiter rein	Technical Onboarding			51
P3	Ablauf des Codes anhand von konkreten Interaktionen mit dem Programm	Technical Onboarding			52
P3	Austausch mit erfahrenen Entwicklern finde ich dabei sehr wichtig	Social Onboarding			53
P3	das durchgucken alter Issues und deren Lösungen finde ich immer sehr hilfreich	Organizational Onboarding			54
P3	Entweder mir fällt selber ein Fehler/Verbesserung auf und ich möchte es umsetzen oder ich sehe Issues anderer Benutzer und möchte diese lösen	Organizational Onboarding			55

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P3	versuche ich mich von oben nach unten durch den Code zu \"wühlen\" um irgendwann an der Stelle anzukommen, an der ich etwas ändern/verbessern möchte	Technical Onboarding			56
P3	verzweige ich oft in die verschiedenen Bereiche des Codes, die an dieser Stelle verwendet werden. So lerne ich nach und nach die Funktionen kennen und wie sie verwendet werden	Technical Onboarding			57
P3	Ich fände ein Beispiel-Issue spannend. Ein Fehlverhalten, das einem präsentiert wird und das man zu lösen hat. Das Beispiel-Issue sollte so gewählt sein, dass man an vielen wichtigen Stellen des Codes vorbeikommt	Ideas on Onboarding Elements			58
P3	bevorzuge doch mehr den direkten Kontakt zu Kollegen/Contributors und das selbständige Entdecken des Codes.	Social Onboarding			59

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P3	Für den allerersten Überblick und eventuell einer fachlichen Vorstellung der Software finde ich spielerische Elemente durchaus angebracht	Attitude towards Play in Onboarding			60
P3	mehr bei den Open Source Projekten sehen, weil hier oft eine direkte und nahe Kommunikation mit anderen Entwicklern nur schwer möglich ist	Contextual Considerations			61
P3	Im Arbeitsumfeld sind Kollegen eher greifbar und bei Fragen ansprechbar	Contextual Considerations			62
P3	Vielleicht mit einem Art Wettbewerb wer einen gewissen Workflow schneller durchgezogen hat	Ideas on Playful Elements for Onboarding			63
P6	but many of the connections were overlapping, hard to see	Experience towards (playful) features			64
P6	line/file size alone does not tell the whole story	Improvements for implemented features			65

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P6	Liked the connections and the overview map, helped me see at a glance what is in the project	Experience towards (playful) features			66
P6	missed actual textual content of the project besides the previews, maybe include more of the contents of the underlying project	Improvements for implemented features			67
P6	Liked the little worm character, felt smooth	Experience towards (playful) features			68
P6	would have liked more connection to the source	Improvements for implemented features			69
P6	Helpful as far as indicating what I have done	Experience towards (playful) features	Companion		70
P6	not much interaction with it, could not click it besides messages	Experience towards (playful) features	Companion		71

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P6	would like to see it on projects of another programming language\nespecially those that maybe are not as cleanly divided into sub projects	Attitude towards Play in Onboarding			72
P6	would have also liked to see more of the structure of the single sub projects	Improvements for implemented features			73
P6	would probably still go at new projects in the IDE itself, seems more efficient at least for the source code	Attitude towards Play in Onboarding			74
P6	I try to get access to everything that I need (git, deployment server, ...) and then set up the project locally and try to build it	Technical Onboarding			75
P6	if there is at any point problems I try to contact the person who wrote the code otherwise the other team members already in the project	Social Onboarding			76
P6	Best case there is documentation for the initial setup, but that is often missing	Organizational Onboarding			77

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P6	try to talk to project management to get to know the organizational structure	Organizational Onboarding			78
P6	Navigation in code I do with IntelliJ and go-to-definition most of the time	Technical Onboarding			79
P6	first step is always the documentation, if there are problems I google or search through the issue list of the package	Technical Onboarding			80
P6	For the source code, would have liked an approach that is near to the code, maybe within the IDE itself and more textual or a little helper within the IDE acting playful	Improvements for implemented features			81
P6	project organization structure maybe a clearer picture of who to talk to and communication with them as part of a game	Improvements for implemented features			82
P6	do not think it can be efficient in day-to-day work	Attitude towards Play in Onboarding			83
P6	maybe for looking at open source projects out of interests	Contextual Considerations			84

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P6	maybe for young developers or people starting with development	Contextual Considerations			85
P6	maybe include more of a progress status	Improvements for implemented features			86
P6	go all the way and create a 3D world from a project that you can go through, although that could be way too far from the project	Ideas on Playful Elements for Onboarding			87
P8	Some code examples listed below looked to be refactor-worthy. For example commented out console logs or very long configurations mapped to string in inline calculations.	Experience towards features	Potential Issue Revealed		88
P8	surprised that even a major project like ethereumjs has the same problems like many other projects like commented out code or console logs or very big util.ts files	Technical Onboarding	Potential Issue Revealed		89

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P8	Some basic information like GitHub stars would be cool at the beginning	Improvements for implemented features			90
P8	I needed some time and the help box to understand, that I was supposed to click somewhere to reveal the information	Experience towards playful features		Also can be seen in the logs, on the overview no help was shown, but then in detail scene it was (line 23471, companion message was closed)	91
P8	I had some problems catching the fast moving dots	Experience towards playful features		Relatively long sequence of reveal clicks without clicking on an element (e.g. json lines 23613-23755)	92
P8	That was fun at first, but at the end it was not that much fun anymore to catch the dots, because it was that easy	Experience towards playful features			93
P8	visualisation of all modules looked a bit chaotic. At first I thought, \"that should help me get a better overview?!\"	Experience towards playful features			94

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P8	It came up when I was the first time in a sub module and didn't know what to do. There is was very helpful	Experience towards (playful) features	Companion	Can be seen on line 23471 - Companion was shown in submodule	95
P8	I don't think that I would prefer such a game/interactive visualisation in contrast to just going throw the filestructure, because the game seemed to be a bit random	Attitude towards Play in Onboarding			96
P8	I don't know if the modules I checked out are the most important ones or not	Improvements for implemented features			97
P8	usually it's a coworker introducing someone new to the project	Social Onboarding			98
P8	I'm checking out a new github project, I just roam around and search for some modules and files whose name seem important to me	Technical Onboarding			99
P8	I just go through the file structure in GitHub and when I wanna dig deeper I open the project in IntelliJ	Technical Onboarding			100

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P8	There could be some moving dots from module to module, depending on how often the module gets imported into another module	Improvements for implemented features			101
P8	recognise which module is used a lot and which not that often	Ideas on Onboarding Elements			102
P8	some activity missing, many games have this discover, fight enemies concept	Ideas on Playful Elements for Onboarding			103
P8	This guide is good for discovering	Experience towards (playful) features	Detail Scene (Within Subpackage)		104
P8	some minigames like fighting the bugs or something similar could make it more interesting to play	Ideas on Playful Elements for Onboarding			105
P8	some videos of the commit history of the project like	Ideas on Playful Elements for Onboarding			106

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P8	In a work environment you are often applied to that project and there is someone who can introduce you	Contextual Considerations			107
P8	In open source project often someone new just joins randomly selfmotivated	Contextual Considerations			108
P8	playful automatic introduction brings more benefits for open source projects, than for work projects	Contextual Considerations			109
P8	find x items of y\ " feature (like the pigeons in gta4)	Ideas on Playful Elements for On-boarding			110
P8	Some speaking character introducing with some basic information	Ideas on Playful Elements for On-boarding			111
P8	Some character to follow, like a rabbit or something that you need to follow from module to module	Ideas on Playful Elements for On-boarding			112

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P8	area constraints, so that you discover a certain area, and after some tasks and knowledge gains you are allowed to discover a broader area	Ideas on Playful Elements for Onboarding			113
P9	damit hab ich schon gearbeitet, hab aber nicht auf den link geklickt	Experience towards (playful) features	Package Revealing	went through pretty fast, not read too much in detail probably	114
P9	grundsätzlich find ichs vertretbar, dass man bei über 1000 zeilen etwas refactoren sollte	Technical Onboarding	Potential Issue Revealing		115
P9	übersicht gut, dass man alle packages auf einem platz sieht	Experience towards (playful) features	Package Revealing		116
P9	mehr info zu den packages gewünscht, zusätzlich zu namen und größe, vielleicht direkt in der übersicht	Improvements for implemented features			117
P9	infos je package ganz gut, dass man die direkt die personen sieht mit dem was sie als letztes gemacht haben	Experience towards (playful) features	Package Revealing		118

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P9	aufdecken war aber bisschen eintönig, da hätte mir was zusätzliches zu machen gefallen	Improvements for implemented features			119
P9	fand die kleine animation mit dem auge ganz gut, aber viel interaktion gab es nicht	Experience towards (playful) features	Companion		120
P9	gut bei neuen projekten grade bei größeren, wo es sonst schwierig is einen überblick zu bekommen \nlinux-kernel würde mir einfallen, also bei großen ganz bekannten projekte, bei denen ich aber nicht unbedingt viel mit der programmiersprache gemacht habe	Contextual Considerations			121
P9	klone mir das projekt als erstes und öffne es dann in meiner ide, da gehe ich dann vom main file ausgehend in die unterfiles, je nachdem was ich machen	Technical Onboarding			122

Part.	Marked Unit	Category Code	Contextual Unit (Feature)	Contextual Unit (Logs)	Nr.
P9	in einer session zusammen die arbeit am projekt anzufangen um direkt einen ansprechpartner für fragen zu haben	Social Onboarding			123
P9	grade als einarbeitung in projekte, wo man schwierig einen überblick bekommen kann	Contextual Considerations			124
P9	vielleicht nicht das schnellste, aber um sich spielerisch eine idee zu verschaffen eine gute idee	Attitude towards Play in Onboarding			125