

universität wien

# MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„HypGen - Generating Testable and Valuable Product Improvement Hypotheses in the Context of Continuous Experimentation“


verfasst von / submitted by

Stefan Berreiter, B.Sc.


angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

| | |
|---|---|
| Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet: | UA 066 926 |
| Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet: | Masterstudium Wirtschaftsinformatik |
| Betreut von / Supervisor: | Univ.-Prof. Dr. Uwe Zdun |

# Acknowledgements

The author would like to thank Jonas Berreiter, Anton La Fontaine, Tobias Fühles, Lukas Lehnert, Michelle Stevic, and Julian Weiß for their valuable support in the context of dry runs and proofreading. Furthermore, thanks go to the case study company and seven product management professionals who participated in the study, often on multiple occasions. And of course, gratitude goes to Dr. Georg Simhandl, who contributed to the success of the Design Science Research with his conceptual feedback and intriguing ideas.

# Abstract

Continuous Experimentation (CEx) is a modern software development paradigm at leading companies such as Microsoft, Airbnb, or Spotify. CEx describes practices and technologies that empower software companies to embed any remarkable product change in an experiment with scientific rigor to reason about its effectiveness. While validating product improvement hypotheses in statistical experiments, companies can rule out product changes that do not improve the software product. Research studies in the field report practitioner insights on how they are conducting CEx at scale and which experimentation systems they have designed. However, current research contributes little to a better understanding of hypotheses generation, i.e. how to best come up with hypotheses to test. This is especially relevant as practitioners report that rejected product hypotheses are rather the norm and not the exception in CEx.

This thesis project applies Design Science Research (DSR) in the course of a case study at a German B2B SaaS company. The DSR goal is to create an artifact that empowers an improved hypotheses generation practice in the context of CEx. In specific, the artifact aims to explore how an effective Decision Support System (DSS) needs to be designed so that it empowers Product Managers to improve their ability to generate (a) statistically testable and (b) valuable hypotheses to improve their software product. To explore the problem space, a comprehensive interview study with product management professionals has been conducted. Additionally, an extensive evaluation was undertaken to evaluate the DSS' capability with regard to empowering improved hypotheses generation.

Results show that HypGen, a DSS for Product Managers, has the potential to improve hypotheses generation. Product Managers found the artifact particularly useful through its workflow approach, integration of experimentation knowledge, target metrics analysis, and feasibility estimation features. In the scope of the case study, cooperating with a hypotheses generation DSS has been found to improve the likelihood of defining product changes as statistically testable hypotheses and to improve the perceived value of product improvement hypotheses. Evaluation has shown that HypGen has guided Product Managers effectively from target metric selection, opportunity identification, and brainstorming product changes to ultimately selecting a feasible experiment design. Moreover, HypGen was perceived to empower a greater adoption of CEx at the case study company. Furthermore, findings indicate that the artifact is applicable in similar B2B SaaS environments.

Results from the interview study and the artifact's evaluation generate knowledge to better understand hypotheses generation and to overcome adoption barriers of CEx. Insights are interpreted to be preeminently relevant for companies that are less mature in CEx, yet are not necessarily restricted to only those.

# Kurzfassung

Continuous Experimentation (CEx) ist ein modernes Paradigma der Softwareentwicklung, das führende Unternehmen wie Microsoft, Airbnb oder auch Spotify anwenden. CEx beschreibt Praktiken und Technologien, die Softwareunternehmen in die Lage versetzen, jede relevante Produktveränderung in einem Experiment hinsichtlich der Frage zu vertesten, ob eine effektive Produktverbesserung wirklich erreicht wurde. Dies erreichen sie mithilfe von Produkthypothesen, die in Experimenten mit wissenschaftlichem Gütegrad getestet werden. Durch die Validierung von Hypothesen in statistischen Experimenten können Unternehmen Produktänderungen ausschließen, die das Softwareprodukt nicht verbessern. Forschungsstudien in diesem Bereich geben meist Einblicke in die Praxis, d.h. wie Unternehmen CEx in großem Maßstab durchführen und welche Experimentiersysteme sie entwickelt haben. Die derzeitige Forschung trägt jedoch kaum zu einem besseren Verständnis der Hypothesengenerierung an sich bei, d.h. wie man am besten Hypothesen zum Testen in Experimenten generiert. Dies ist besonders wichtig, da Praktiker berichten, dass abgelehnte Produkthypothesen bei CEx eher die Regel als die Ausnahme sind.

Dieses Master Thesis Projekt wendet Design Science Research (DSR) im Rahmen einer Fallstudie bei einem deutschen B2B SaaS-Unternehmen an. Das Ziel dieses DSR ist es, ein Artefakt zu erstellen, das eine verbesserte Hypothesengenerierungspraxis im Kontext von CEx ermöglicht. Konkret zielt das Artefakt darauf ab, zu erforschen, wie ein effektives Decision Support System (DSS) gestaltet sein muss, damit es Produktmanager in die Lage versetzt, ihre Fähigkeit in zwei Belangen zu verbessern. Das DSS soll sowohl dabei helfen (a) statistisch testbare als auch (b) wertvolle Hypothesen zur Verbesserung des Softwareprodukts zu generieren. Um das Problemfeld zu erforschen, wurde eine umfangreiche Interviewstudie mit Produktmanagement-Experten durchgeführt. Zusätzlich wurde eine aufwändige Evaluierung durchgeführt, um die Fähigkeit des DSS zur verbesserten Hypothesengenerierung zu bewerten.

Die Ergebnisse zeigen, dass HypGen, ein DSS für Produktmanager, das Potential hat, Hypothesengenerierung zu verbessern. Produktmanager fanden das Artefakt besonders nützlich durch seinen Workflow-Ansatz, die Integration von Experimentation Wissen, die Analyse von Zielmetriken und die Funktionen zur Abschätzung der Hypothesentest-Machbarkeit. Im Rahmen der Fallstudie wurde festgestellt, dass die Zusammenarbeit mit einem DSS zur Hypothesengenerierung die Wahrscheinlichkeit erhöht, Produktänderungen als statistisch testbare Hypothesen zu definieren und den Wert der generierten Produktverbesserungshypothesen zu steigern. Die Evaluierung hat gezeigt, dass HypGen die untersuchten Produktmanager effektiv von der Auswahl der Zielmetriken, der Identifizierung von Produktverbesserungsmöglichkeiten über das Brainstorming von Produktänderungen bis hin zur Auswahl eines durchführbaren Experiments führen konnte. Darüber hinaus wurde HypGen als für die breitere Anwendung von CEx förderlich empfunden, zumindest

im Falle der Case Study Firma. Die Ergebnisse deuten auch darauf hin, dass das Artefakt in ähnlichen B2B SaaS-Umgebungen anwendbar ist.

Die Ergebnisse der Interviewstudie und die Auswertung des Artefakts generieren Wissen, um das Phänomen Hypothesengenerierung besser zu verstehen und um Barrieren bei der Nutzung von CEx zu überwinden. Die Erkenntnisse werden so eingeschätzt, dass sie vor allem für Unternehmen relevant sind, die auf einem niedrigeren Reifegrad im Bereich CEx sind – doch sind die Ergebnisse nicht unbedingt nur auf diese beschränkt.

# Contents

**Acknowledgements** i

**Abstract** iii

**Kurzfassung** v

**List of Tables** xi

**List of Figures** xiii

**Listings** xv

**1. Introduction** 1

**2. Research Field** 3
2.1. Continuous Experimentation . . . 3
2.1.1. Paradigm . . . 3
2.1.2. Experiment Designs . . . 4
2.1.3. Implementation of Experimentation . . . 5
2.2. Hypotheses Generation . . . 6
2.2.1. Research Landscape . . . 6
2.2.2. Statistical Hypotheses Testing . . . 6
2.3. Motivation for Thesis Research . . . 7
2.3.1. Understudied Phenomenon . . . 7
2.3.2. Problem Space in Focus . . . 8

**3. Design Science Research** 9
3.1. Theoretical Foundation . . . 9
3.1.1. The Design Science Paradigm . . . 9
3.1.2. Theoretical Cornerstones . . . 9
3.1.3. Practical Execution . . . 11
3.2. Thesis Project . . . 11
3.2.1. Case Study Environment . . . 11
3.2.2. Research Objective . . . 12
3.2.3. Research Process . . . 13

**4. Explorative Research** **15**
4.1. Approach . . . 15
4.1.1. Research Method . . . 16
4.1.2. Sample . . . 17
4.1.3. Data Analysis . . . 17
4.2. Results . . . 18
4.2.1. Descriptive Findings About Interviewees . . . 18
4.2.2. Process Perspective . . . 19
4.2.3. Concepts and Artifacts . . . 24
4.2.4. Rules, Heuristics, and Intuition . . . 30
4.2.5. Product Management Tools . . . 33
4.2.6. Reflection on Status-Quo . . . 35
4.3. Discussion . . . 37
4.3.1. Condensation of Results . . . 37
4.3.2. Generated Design Knowledge . . . 39
4.4. Artifact Design Objectives . . . 41

**5. Foundational Theories & Methods** **43**
5.1. Decision Support Systems . . . 43
5.1.1. Overview . . . 43
5.1.2. Leveraged Design Knowledge . . . 44
5.2. Experimentation Platforms . . . 45
5.2.1. Overview . . . 45
5.2.2. Leveraged Design Knowledge . . . 46
5.3. Hypotheses Centered Decision-Making . . . 48
5.3.1. Overview . . . 48
5.3.2. Leveraged Design Knowledge . . . 49
5.4. Product Analytics . . . 50
5.4.1. Overview . . . 50
5.4.2. Leveraged Design Knowledge . . . 51
5.5. Product Metrics Frameworks . . . 52
5.5.1. Overview . . . 52
5.5.2. Leveraged Design Knowledge . . . 54

**6. Artifact Design** **55**
6.1. The Artifact - HypGen . . . 55
6.1.1. Core Idea . . . 55
6.1.2. Final Design . . . 56
6.1.3. Intermediate Designs . . . 63
6.2. System Design . . . 67
6.2.1. Tech Stack . . . 67
6.2.2. Scenarios . . . 69
6.2.3. Logical View . . . 70
6.2.4. Process View . . . 77

6.2.5. Development View . . . 79
6.2.6. Deployment View . . . 81

**7. Evaluation** **83**
7.1. Method . . . 83
7.1.1. Evaluation Strategy . . . 83
7.1.2. Evaluation Criteria . . . 84
7.1.3. Execution . . . 85
7.1.4. Analysis . . . 87
7.2. Results . . . 88
7.2.1. Ex Ante Insights . . . 88
7.2.2. Overall Research Question . . . 89
7.2.3. Artifact's Evaluation Criteria . . . 90
7.2.4. Investigation of Core Value . . . 93
7.2.5. Task Performance . . . 95
7.3. Summary . . . 102

**8. Conclusion** **105**
8.1. Contributions . . . 105
8.1.1. Relevance in Application Domain . . . 105
8.1.2. Addition to Knowledge Base . . . 105
8.1.3. Implications for Ongoing Projects . . . 110
8.2. Review of Design Science Research Guidelines . . . 111
8.3. Limitations . . . 113
8.4. Avenues for Future Research . . . 115

**Bibliography** **117**

**A. Appendix** **133**
A.1. Implementation Project . . . 134
A.2. Author's Affiliation with the Case Study Company . . . 135
A.3. Author's Blogging about CEx . . . 136
A.4. Dry Runs & Support . . . 137
A.4.1. Pre-Study Dry Run . . . 137
A.4.2. Pre-Study External Labeler . . . 137
A.4.3. Evaluation Dry Run . . . 138
A.5. Semi-Structured Interview Guideline . . . 139
A.5.1. Onboarding . . . 139
A.5.2. First Block . . . 139
A.5.3. Second Block . . . 140
A.5.4. Third Block . . . 140
A.5.5. Control Questions . . . 141
A.5.6. Outro . . . 141
A.6. Roles Mentioned in Decision-Making Process . . . 142

A.7. Product Manager Tool Landscape . . . . . 143
A.8. Main Architectural Design Decisions . . . . . 144
A.9. Final Evaluation Questions . . . . . 146
A.10.Areas of Interest Overview . . . . . 148
A.11.Event and Survey Data . . . . . 151

# List of Tables

4.1. Role-Specific Experience of Interviewees . . . . . 18
4.2. Educational Background of Interviewees . . . . . 19

7.1. Artifact's Evaluation Criteria . . . . . 85
7.2. Evaluation Subjects . . . . . 86

8.1. Satisfaction of DSR Guidelines. Adapted from [1] . . . . . 113

A.1. Roles Within The Product Management Process . . . . . 142
A.2. Current Product Manager Tool Landscape . . . . . 143
A.3. Evaluation Survey Questions . . . . . 147

# List of Figures


3.1. Design Science Research Framework [1] . . . 10
3.2. Components of Design Knowledge [2] . . . 11
3.3. Design Science Research Methodology Process [3] . . . 12

5.1. Data Mining Enhanced IDSS for Better Decision Support [4] . . . 44
5.2. Metrics Catalog at Spotify [5] . . . 46
5.3. Sample Size Estimations at Spotify [5] . . . 47
5.4. GoodUI.org's Product Improvement Patterns [6] . . . 53

6.1. Extended Hypothesis Definition . . . 56
6.2. Design: Overall User Interface . . . 57
6.3. Design: Create Product Area . . . 57
6.4. Design: Configure Health Metrics . . . 58
6.5. Design: Improvement Opportunities . . . 59
6.6. Design: Hypothesis . . . 60
6.7. Design: Product Change Suggestions . . . 61
6.8. Design: Experimentation . . . 62
6.9. Design: Outcome . . . 62
6.10. Design: Hypotheses Roadmap . . . 63
6.11. v1: Yaml-Spec and CLI . . . 63
6.12. v1: User-Interface (read-only) . . . 64
6.13. v2: Configurable Health Analyses for Custom Product Area . . . 65
6.14. v2: Revamped Opportunity Ranking . . . 65
6.15. v2: Hypothesis Template and Product Change Suggestions . . . 66
6.16. HypGen's Tech Stack . . . 67
6.17. Use Case Diagram . . . 69
6.18. Class Diagram: Layered Architecture Kernel . . . 71
6.19. Class Diagram: Shared Kernel . . . 72
6.20. Class Diagram: Product Admin . . . 72
6.21. Class Diagram: Product Analytics . . . 73
6.22. Class Diagram: Experimentation . . . 74
6.23. Class Diagram: Validation Store Detail . . . 75
6.24. Class Diagram: Suggestions . . . 76
6.25. Sequence Diagram - General Communication . . . 77
6.26. Sequence Diagram - Cross Bounded Context Communication . . . 78
6.27. Component Diagram . . . 80
6.28. Deployment Diagram . . . 81

7.1. Subjects' Self-Reported Application Frequencies . . . . . 88
7.2. Overall Research Evaluation . . . . . 90
7.3. HypeGen Recommendation . . . . . 91
7.4. Artifact's Evaluation Criteria . . . . . 92
7.5. HypeGen Capabilities and Their Perceived Value . . . . . 93
7.6. Dwell Time in Different Areas of Interest . . . . . 95
7.7. Ease of Task Completion . . . . . 96
7.8. Time to Task Completion . . . . . 97
7.9. Interaction Plot: Task 1 Exploration vs. Exploitation . . . . . 98
7.10. Interaction Plot: Task 2 . . . . . 99
7.11. Mouse Heatmap: Task 3 (Subject S5) . . . . . 100
7.12. Interaction Plot: Task 4 . . . . . 101
7.13. Interaction Plot: Task 4 Exploration vs. Exploitation . . . . . 101
7.14. Interaction Plot: Task 5 . . . . . 102

8.1. beyond.so Public Beta . . . . . 111

A.1. Medium Articles on Experimentation . . . . . 136
A.2. AoIs - Product Area . . . . . 148
A.3. AoIs - Product Health (HEART) . . . . . 148
A.4. AoIs - Improvement Opportunities . . . . . 149
A.5. AoIs - Hypothesis and Suggestions . . . . . 149
A.6. AoIs - Experimentation . . . . . 150
A.7. AoIs - Hypothesis Implementation & Outcome . . . . . 150

# Listings

5.1. Event Example . . . . . 51

A.1. Event Example (HypGen) . . . . . 151
A.2. Survey Response Example (HypGen) . . . . . 151

# 1. Introduction

For a software product to be well received by its customers, the right features with the right experience must be offered. Since software today is usually developed and rolled out in increments in an agile manner (cf. [7]), the quality of the software product is ultimately the sum of everyday product decisions. Unfortunately distinguishing between good and bad product changes is not straightforward, so companies aim for validated learnings about the effectiveness of potential product improvements. In an ideal world, this ensures that only product improvements and no degradations are implemented. To achieve this, leading companies increasingly apply Continuous Experimentation (CEx) which enables a process where 'developers treat every planned feature as an experiment' [8]. This means that product changes are translated into statistically testable hypotheses that are evaluated in experiments. Finally, approved or rejected hypotheses then influence a product change's rollout or rollback decision. Following this logic, usually only winning hypotheses and hence effective product changes should become part of the software product.

Scientific reviews, case studies, and practitioner blog posts demonstrate well how CEx at scale works at companies like Microsoft [9, 10], Netflix [11], Airbnb [12], Meta [13], or Spotify [14, 5]. Moreover, statistical aspects of hypotheses testing in CEx have been discussed thoroughly (cf. [15]). However both, academic and gray literature provide little insight into how hypotheses are initially generated. Ironically, practitioners report that experiments reveal that hypotheses aimed to improve a software product are much more often wrong than right, with hypothesis rejection rates of up to 90% in mature products [16]. To shed light on hypotheses generation, the emerging research domain *Hypotheses Engineering* has been proposed [17] where first papers gave an overview of the research landscape [18]. While many domains were found to be influenced by *Lean Startup* [19] ideas to work hypotheses-driven, insights reveal that hypotheses generation in the context of making feature decisions in CEx is currently not well understood [18]. In support of that, Schermann et al. conclude in their study on CEx practices that decision-making about which features to develop and in which parts of the software product to experiment is still often delegated to product decision makers' intuition [20]. Besides improvement potential in practices surrounding hypotheses generation, Schermann et al. also identify major problems of leveraging CEx which further motivate for creative solutions to target the CEx adoption problem.

To address the knowledge gap regarding hypotheses generation and to contribute to greater adoption of CEx, this thesis is applying Design Science Research (DSR) to create and evaluate a Decision Support System (DSS). The DSS aims at supporting Product Managers in generating product improvement hypotheses to be validated in the course of CEx. The research study investigates the decision-making of product management

professionals at a German B2B SaaS company. These professionals – mostly Product Managers – are observed to learn how they come up with assumptions or hypotheses on which product changes to develop. Based on a deep understanding on Product Managers' decision-making processes, a DSS is iteratively created and evaluated to answer the overall research question: *'How should a Decision Support System be designed to improve the generation of testable and valuable product improvement hypotheses in a SaaS startup context'*?

The thesis paper[1] is structured as follows. In chapter 2, an overview of relevant literature to motivate the problem area is provided. Then, chapter 3 continues with an introduction of Design Science Research in general and the peculiarities of this research. In chapter 4, an extensive interview pre-study with product management professionals is presented that leads to solution objectives for the subsequently designed DSS. Following in chapter 5, literature addressing similar topics and design challenges are introduced that have inspired artifact development. In chapter 6, the designed artifact is demonstrated from a user as well as from a technical perspective. Continuing in chapter 7, the final evaluation is presented to assess if the designed artifact has successfully addressed its solution objectives. In chapter 8, the paper concludes with key knowledge contributions while also presenting limitations and opportunities for future research.

[1]The thesis partly includes literature research from the preceding Master's Seminar. This is made transparent when presenting the current research field in chapter 2 as well as literature findings in section 5.2. In addition, the section A.3 also describes two blog posts that the author has written in this area to ensure the delineation of the Master's Thesis from other work of the author.

# 2. Research Field

Before presenting the empirical thesis project, the author introduces the relevant literature on Continuous Experimentation (CEx) and hypotheses generation. These sections summarize content already covered by the author in the preceding Master's Seminar. Furthermore, it is presented how the thesis project addresses an underrepresented aspect of the literature and thus motivates the research project.

## 2.1. Continuous Experimentation

### 2.1.1. Paradigm

The field of Continuous Experimentation (CEx) is a collection of methods and technologies to test changes in software behavior or infrastructure in controlled experiments in order to decide if a certain change should become a stable part of the software. Thus, CEx's target is to define every noteworthy software change as a verifiable hypothesis and to subsequently test it in an experiment [21]. The naming CEx has been introduced by Etsy's Dan McKinley in 2012 [22] and has been taken over by authors such as Fagerholm thereafter [21]. CEx can be seen in very close relation to related terms found in the literature such as *online controlled experiments* (as used by e.g. Kohavi [9]) and *field experiments* (as used by e.g. Mattos [23]). However, CEx's focus on the *continuous* aspect conveys the modern tendencies of experimentation even better, i.e. to make experimentation an integral part of *continuous software engineering*[1] [24]. This naturally docks experimentation more systematically with infrastructural aspects (e.g. [21]) and ensures that experimentation is systematically leveraged, not just on a few occasions. To conclude, CEx describes the wrapping of any remarkable product changes into testable hypotheses and rolling out these hypothesis-wrapped product changes in experimental designs with scientific rigor to later reason if the change actually improved the product.

For years, scientific literature on CEx and related fields has gained increasing attention. Already in the 1980s, Basili et al. defined a framework for conducting controlled experiments [25] in software settings. In the late 2000s and simultaneous to the advent of web applications, practitioners from leading research departments in Silicon Valley started to share their insights in scientific papers. As one example, Kohavi[2] wrote an influential article in 2009 [15] and co-authored one further [26] from a practitioner perspective which encouraged other authors to further research. Since then, highly cited authors such as Olsson and Bosch [27, 28] and lately Fagerholm [21, 29], Schermann [20, 30], and

[1] With practices such as Continuous Integration and Continuous Delivery/Deployment (CI/CD)
[2] Back then establishing CEx at Microsoft

Fabijan [31, 32] contributed renowned articles to the field. The research field also received extensive literature reviews that map the landscape more extensively [33, 23, 34, 35].

### 2.1.2. Experiment Designs

Experiments in which product hypotheses are tested can be considered to fall under one of two types. As software applications can be considered a socio-technical system [36], the two parts of the dichotomy (i.e. users and technology) can be the subject of an experiment. Scientists refer to primarily user-behavior-driven experiments as *business-driven* and to technological-related experiments as *regression-driven* experiments [20]. In business-driven experiments, a change in the product is hypothesized to have a positive effect on product usage in some form that is relevant to the business. On the other hand, regression-driven experiments mainly investigate product changes with respect to technical performance effects and mostly serve as quality assurance technique. While the two types of experiments might be perceived to be mutually exclusive, they often tend to overlap. As technical changes can impact user metrics - and changes meant for improving user experience might have technical side-effects, it is suggested to control for unwanted side-effects with the help of guardrail metrics [5].

While the dichotomy between business-driven and regression-driven experiments serves as guidance, it says little about the way an experiment is conducted. Experiments as reported in the literature mostly fall under the category of randomized controlled experiments, however, also other approaches have been reported. As the classic CEx experiment design, randomized controlled experiments in its usual form distribute users into two groups (A/B) randomly and provide one group with the treatment (the new feature or bug fix) and the control group with the existing version [15]. This experiment setup necessitates a random assignment of users or traffic into two groups and is especially suited to detect a change's standalone effect, which is why it is also referred to as the gold standard [37]. Second, multi-armed bandit experiments are used where multiple competing variants are continuously benchmarked to each other with the better variants prevailing over time automatically [38, 39]. Third, quasi-experiments can be run while assigning users in groups based on common characteristics (e.g. web browser, geography), therefore not randomly [40, 41]. As characteristics themselves often have a significant effect on a feature's performance (as can be seen in [42]), this approach has some drawbacks. Fourth, as reported by Schermann et al. [20] when conducting regression-driven experiments, users are not necessarily assigned to different groups, so the experiment consists of informally validating changes between the two variants of technology (e.g. old database vs. new database) and not changes between users in two groups.

Furthermore, there exist three more variants that are worth mentioning. First, qualitative experiments describe set-ups where hypotheses are validated with the help of qualitative interviews and user testing methods [43], which in turn informs the creation of features. Second, multi-factorial or multi-variate tests may be applied where the product is changed along a few dimensions (e.g. call to action, button color, and image changed at the same time), and all possible combinations are tested against each other. However, literature tends to not consider it a good practice in the area of CEx [9]. Third, crossover

experiments - i.e. studying a user's change of behavior over time with the introduction of different features - could potentially also be run as described by Mattos [23]. However, it seems that CEx practitioners do not significantly leverage these three further approaches, providing further support to the relevance of classic A/B testing.

### 2.1.3. Implementation of Experimentation

Similar to lab experiments, software experiments cannot take place in the void and hence need to have technological components in place. Especially business-driven experiments need to distinguish between two or more groups of users or requests thereof. For randomized controlled experiments, randomization algorithms are used to determine which group a user belongs to either based on rudimentary hashing of user or session IDs [44] or more sophisticated bucketing approaches [45]. A group-specific marker is set to e.g. the request's header and the software should respond with different behavior [44], i.e. showing the product change (treatment) or the existing solution (control) in an A/B test. This can be done either with conditional code paths based on the group assignment, i.e. a feature flag [44], or via the execution of a different code configuration on another system than the code itself [13]. If not only the (mostly business-driven) behavior of one system should be checked, dynamic traffic routing [20, 30] can direct users to one of many servers where different versions of the product are installed. All approaches include checking the group of a user (request) and then deciding per-group behavioral aspects such as code execution, routing of requests - or even user interface styling [14].

To analyze experiments and decide whether a change is positive, negative, or likely neutral, practitioners rely on statistical methods to accept or reject the product improvement hypothesis. When it comes to statistics, practitioners rely on either Frequentist or Bayesian statistics [46]. Frequentist statistics solely use data from within the experiment while applying Bayesian statistics spots differences while factoring in collected data in the past to calculate prior probabilities [46]. Whereas the Bayesian approach is less intuitive and drawbacks such as peaking need to be considered (i.e. problems from stopping the experiment too early), it also has upsides like usually smaller sample sizes needed to receive significant results. On the other hand, Frequentist approaches make it easy to determine the minimum sample size upfront to run an experiment. Moreover, the Frequentist paradigm is usually more commonly found in tool providers on the market, although on the cost of experiments powered by Frequentist statistics running usually for longer than Bayesian ones [47].

CEx practitioners are not setting up and evaluating experiments by hand, but they are leveraging dedicated experimentation platforms. A variety of off-the-shelf SaaS solutions for experimentation exists such as Optimizely [48], Kameleoon [49] or PostHog [50]. Also, industry-leading companies have built and reported on their own systems such as Spotify [14], Airbnb [12] or Microsoft [10]. While home-grown or off-the-shelf solutions are different, these systems usually support the same core functionalities. These involve planning the experiment with an already existing hypothesis, handling random assignments of users into groups, managing feature flags for showing different functionalities to different user groups and offering capabilities for experiment analysis. Especially to satisfy more complex

requirements like concurrent experimentation [45] and to achieve their own flavors of experimentation (like sequential testing at Spotify [5]), large enterprises build their own experimentation platforms[3].

## 2.2. Hypotheses Generation

### 2.2.1. Research Landscape

All experiments start with a hypothesis. The reason is that a hypothesis is a prediction that by some intervention (i.e. a certain product change), user and/or system behavior is meant to be improved. If there was an improvement associated with the product change, this would need to be unambiguously assessed by analyzing one or more key performance indicators. Because only a few scholars have concentrated on the creation of hypotheses, Melegati et al. propose a new research field called *Hypotheses Engineering* [17] which focuses on understanding how hypotheses are designed. As literature is described as scarce [17, 51], Melegati et al. have conducted a subsequent gray literature survey which revealed all sorts of different hypotheses generation techniques. They found in their review that literature on hypotheses generation is heavily influenced by *Lean Startup*'s ideas [19] and is more focused on validating assumptions of business models rather than assumptions about product changes in the context of CEx [18]. Moreover, they find that formalizations of hypotheses only exist in the form of templates where parts of sentences are filled in, while some templates build the bridge to measurement of the experiments. In support of that, another study finds that startups are often not particularly adept at defining hypotheses for experimentation which makes hypotheses testing ambiguous and does create rather vague learnings [51]. To date, few projects to support hypotheses generation have been undertaken in this area. One noteworthy example is Melegati et al.'s [52] work on *HyMap* in 2022. They present a cognitive mapping process aimed to help company founders with the help of a facilitator to come up with hypotheses based on a set of predefined questions. However, it seems technological support and a close connection to CEx is currently not well explored.

### 2.2.2. Statistical Hypotheses Testing

Hypotheses do not simply exist, they must be defined by product specialists usually looking at mismatches of expectations and reality. The rationale is that by looking at surprisingly good or bad results, insights about these gaps can lead to new ideas on how to improve a software (i.e. which product change to conduct). For instance, in Olsson's HYPEX model [27], strategy is analyzed and should lead to the generation of features which in turn inspires assumptions about features' measurable improvements. After the development of a feature, a gap analysis is undertaken to compare desired and real changes in feature behavior, which ultimately informs the generation of hypotheses. While experimentation systems matured in rolling out the hypotheses-wrapped features

[3]More on experimentation platforms can be found in chapter 5

in experiment settings in a highly automated fashion - hypotheses generation is mainly delegated to product management professionals and described as rather un-digitized[4]. Major activities are attributed to product decision-makers' understanding of strategy and business needs without stating exact practices further. To conclude, literature only provides a rudimentary understanding of how product hypotheses for CEx are defined.

In order to reject or approve a hypothesis, the hypothesis needs to be tested in an experiment that yields enough statistical power. Only with enough power, results from the group experiencing the product change can be extrapolated to the whole application's user audience. In order to drive statistical power, the confidence level to detect significant deviations between control and treatment needs to be taken into consideration. Scientists and practitioners alike conclude a 95% confidence interval seems to be the best fit [53]. Obviously, the power can also be improved by testing the experiment on a larger sample size [53], however, this always comes with the risk of exposing more users to a potentially worse option [15]. Also, the target metrics' variance needs to be factored in, the lower the variance, the easier it is to detect significant changes. Likewise, the effect size also referred to as the Minimum Detectable Effect (MDE) the treatment is hypothesized to have on key metrics *at least*, needs to be carefully considered [14, 53]. Larger effect sizes lead to lower sample sizes for experimentation, however at the cost of more frequently rejecting product hypotheses that actually have been effective. To conclude with a rule of thumb, 'a 5% false positive rate and targeting 80% power are well-established conventions that balance between limiting false discovery and enabling true discovery' [53].

## 2.3. Motivation for Thesis Research

### 2.3.1. Understudied Phenomenon

Once looking at highly-cited authors and practitioner success reports, it could be perceived as if CEx is an established practice throughout the industry. Many authors report on their large-scale experimentation efforts at leading companies such as Microsoft, AirBnb, Spotify, or Meta. However, they do so with a strong focus on experimentation with preexisting hypotheses that they have rejected or approved in the course of a thought-through experiment design with a large user base. However, it has been previously noted that hypotheses generation is largely delegated to product professionals and their intuition. Further, from looking at industry leaders alone, is not apparent that the ease of CEx adoption is also true for other software companies.

Investigating non-industry leaders yields a more nuanced perspective on CEx. The reason is that for companies with different constraints, the business case for CEx is not that straightforward and easily accepted [20]. To support this, Schermann et al.'s study shows that a variety of reasons such as inflexible software architecture, low number of users to test with, general doubt of experimentation's added value, and lack of experimentation expertise are pointed to as barriers to CEx adoption. Mantyla et al. [54] show in their multiple-case study that CEx was not really applied in the software startups investigated.

[4] cf. many examples of templates or the empirical results in Schermann et al. [20]

These studies show that CEx might not just be a *plug-and-play* solution that can be easily adopted and moves a company up the stairway to heaven, i.e. to reach a high maturity in automated experimentation-driven software development [55]. This notion is strengthened by industry leaders looking at other companies. As Intuit's experimentation leader Lucian Lita put it: 'For all the talk around data and experimentation, most companies that I've interacted with, to this day don't use A/B testing, but rather rely solely on gut feel and intuition to make product decisions' [56]. Also, Schermann et al. note that *which* parts of the software to improve (i.e. *where* to generate hypotheses to solve problems) and *which* metrics to select (i.e. how to determine the positive effect of a product change), is largely delegated to intuition, informal past experience, and individual talent of people.

### 2.3.2. Problem Space in Focus

The thesis research aims at the primary objective to improve hypotheses generation in the context of CEx efforts. Once hypotheses generation is supported effectively, the research also aims at contributing towards greater adoption of CEx. In particular, the study wants to find out how an effective solution must be designed to support the generation of valuable statistically testable product hypotheses. If successful, such a system can be integrated with experimentation systems and can pave the way for CEx to be adopted at a larger scale in SaaS startup environments.

> 'It raises the question to what extent our view of these practices is coined by the peculiarities and needs of a few innovation leaders, such as Microsoft, Facebook, or Google.' [20]

Because many reports focus on case studies that shed light on US companies, the author conducts this thesis study at a German SaaS company that does not systematically leverage CEx. While learnings from a case study cannot be transferred to the entire industry without hesitation, this thesis research can certainly contribute more nuanced learnings for many companies and industries. In specific, this research's implications might be applicable for those companies and industries that either have hurdles in adopting CEx - or at least are very intuition-driven in their hypotheses generation and metrics selection. To keep the scope manageable, the study focuses on business-driven experiments and hence omits regression-driven experiments.

# 3. Design Science Research

In this chapter, an overview of Design Science Research (DSR) is presented. Based on an understanding of the overall methodology, the specifics of this DSR project are illustrated.

## 3.1. Theoretical Foundation

### 3.1.1. The Design Science Paradigm

Design Science Research (DSR) is a complementary research paradigm to behavioral science and is especially useful in Information Systems research. In comparison to behavioral research, DSR does not only describe the status quo of phenomena in a certain domain. But, DSR builds on the development of an artifact that generates knowledge by finding out how potential solutions can solve problems in the investigated domain. Hevner et al. illustrate differences and the complementary nature in a simple four-sentence logic: 'The goal of behavioral science research is truth. The goal of design science research is utility. [..O]ur position is that truth and utility are inseparable. Truth informs design and utility informs theory' [1]. Hence, by developing a solution to an identified problem and evaluating its usefulness in solving the problem, empirical knowledge can be generated that goes beyond the observation of the existing domain (cf. Brocke et al. [57]).

Beyond solving real-world problems, a core necessity of DSR is that design knowledge has to be generated and communicated properly. Identifying, evaluating, and interpreting effective design approaches yield insights that can be added to the body of knowledge. These approaches distinguish DSR from 'routine design' where real-life problems are simply solved by solely leveraging already existing knowledge [57]. Various research that builds on DSR has been conducted to date with a core criticism that many research projects rather stand on their own than being effectively embedded in the greater DSR knowledge landscape [2]. Therefore, the embedding and grounding of the individual project into the greater research landscape is of high importance to guarantee rich knowledge contributions to the field.

### 3.1.2. Theoretical Cornerstones

Various frameworks describe the core idea of DSR from different angles, yet they aim at an overarching narrative of iteratively developing and evaluating solutions to an identified problem. The underlying commonality is that these frameworks always feature a problem space and a solution space. By means of developing and evaluating the effectiveness of a solution (i.e. the *artifact* in DSR terms) for relevant problems within the problem space, the researcher creates knowledge about a design's utility. This prescriptive knowledge [2]

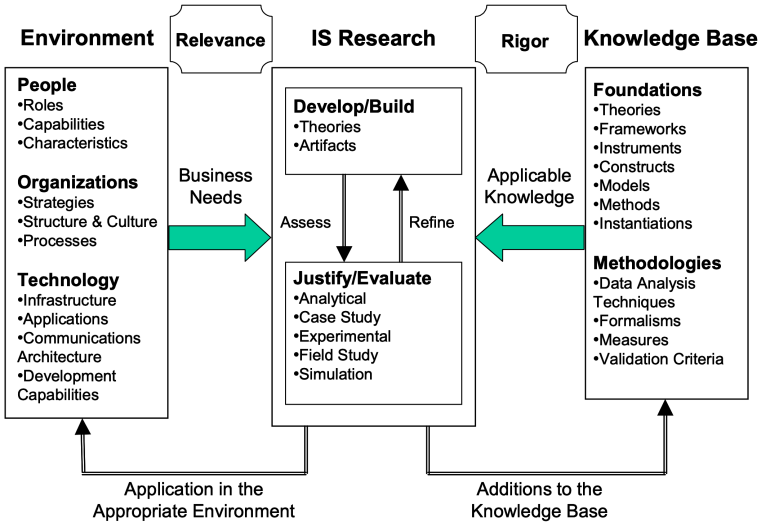



Figure 3.1.: Design Science Research Framework [1]

is then added to the scientific knowledge stack in the form of a knowledge contribution. Hevner et al. [1] frame three different areas within DSR that the researcher interacts with during the course of the complex research process (cf. Figure 3.1). The idea of the *Environment* describes the area where problems originate, including their people, interactions as well as technologies. *Information Systems Research* creates the solutions to solve these problems. To assess the degree of problem-solving, Information Systems Research also offers a variety of scientific methods. To continue, the researcher does not work in void nor start from scratch, but the researcher both leverages the existing *Knowledge Base* about similar problems or solution approaches and in turn also contributes generated knowledge to the base again.

While viewing through a design knowledge lense, Brocke et al. [2] contribute a similar notion to better understand DSR. Through their lens, design knowledge is produced via the application and evaluation of a solution (from the solution space) to solving a problem in the problem space (cf. Figure 3.2) to reason about the utility of a certain solution approach. While the latter is more simplistic in nature, the two perspectives together describe a good overview of the general foundation of DSR.

Within this context of solving problems and extending knowledge, the DSR scholar operates in three cycles according to Hevner [58]. The researcher works in the *Relevance* cycle to motivate the need for a solution through the identification of a noteworthy problem. In the *Design* cycle, the researcher iteratively develops the artifact and approaches a capable solution for the problem while mostly factoring in feedback from the people facing the problem. While in the *Rigor* cycle, the researcher aims at effectively applying previous theories and methodologies as well as communicating knowledge back [58]. All of these cycles are not sequential in nature, but they intertwine and guide the researcher towards an effective DSR project during the course of a few design cycles.

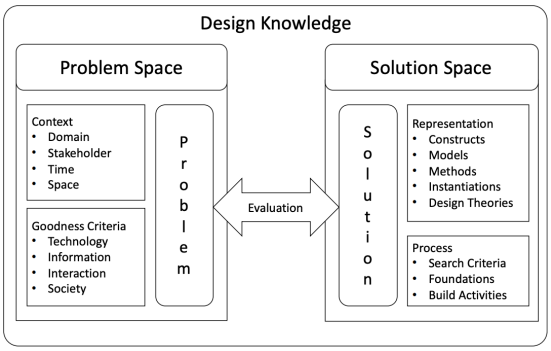



Figure 3.2.: Components of Design Knowledge [2]

### 3.1.3. Practical Execution

Beyond general foundations, researchers have come up with process frameworks to guide the practical execution of a DSR project. Peffers et al. contribute a widely cited process framework on how to conduct a DSR project (cf. Figure 3.3). Starting with one of four possible research entry points, an iterative process to ultimately shape the design artifact and communicate knowledge is created. Following Peffers et al. [3], usually by means of an investigation of the environment, the problem space, an important problem is identified. The problem in turn motivates the objective for the search for an effective solution (i.e. *Problem-Centered Initiation* in Figure 3.3). Once having collected enough knowledge about the problem domain, the problem is translated into objectives that the solution should satisfy. Following from there, the design is iteratively developed, demonstrated, evaluated, and communicated - before entering the next process iteration (often referred to as build and evaluate loop). The newly generated knowledge stemming from an intermediate design and its evaluation is then incorporated into the next iteration. As the goal is not only to solve the problem but to contribute knowledge (even on intermediate design choices and their effectiveness), the build and evaluate loops should ensure a rich communication of the gained insights towards each cycle's end.

## 3.2. Thesis Project

After a general overview of Design Science Research (DSR) and its intertwined process, the thesis DSR project can now be presented in greater detail.

### 3.2.1. Case Study Environment

To start with, DSR scholars advise that the problem to be solved should be properly positioned, i.e. which parts of the problem and solution space are addressed [2]. In chapter 2, the author has already provided an overview of the literature on CEx and hypotheses generation which yielded a research gap that motivates this DSR project.

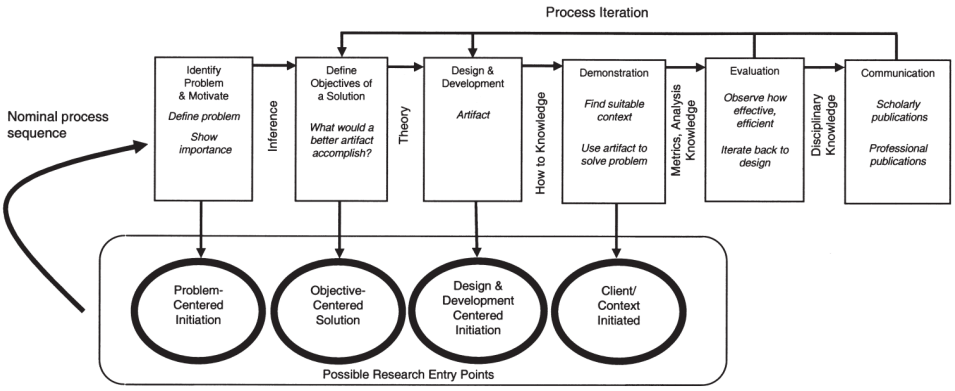



Figure 3.3.: Design Science Research Methodology Process [3]

To ensure in-depth findings as well as an evaluation of the artifact in a realistic setting, the DSR projects takes place within a case study setting. The case study company is a B2B SaaS firm offering a software product to business clients with a total user base between 10k - 50k users. The company makes use of a yearly subscription model to access its software product. Business customers buy or renew user subscriptions for their employees, the users of the software. The company has been selected due to two reasons. First, the company is operating in a research-driven and iterative way of product development, however does not leverage CEx systematically. This constitutes a case that can contribute learnings to the literature which is heavily dominated by reports on big industry leaders. Additionally, it is a case where learnings about driving CEx adoption with the help of improved hypotheses generation can be generated. Second, due to the author's affiliation with the company, ease of access to important stakeholders and access to documents is possible. Whilst having considerably good access for a researcher, the author undertakes several measures for unprejudiced scientific investigation in the course of this thesis project[1].

The company's core product is a SaaS application that helps its clients with a variety of functionality to support the execution of strategic projects. From a superficial technical perspective, their product consists of a monolithic backend and a single-page application frontend, both hosted in the cloud. The company accompanies its software product sales with a small number of professional service specialists who empower their business clients to work effectively with its software product.

### 3.2.2. Research Objective

The main objective of the research can be summarized best as *'How should a Decision Support System be designed to improve the generation of testable and valuable product improvement hypotheses in a SaaS startup context'?* To elaborate further, the goal is to

[1]More detail can be found in section A.2 and when discussing research limitations in chapter 8

generate knowledge about an effective software design that is helping Product Managers to translate product changes into valuable, statistically testable product hypotheses. Product Managers at the B2B company are the key people involved in making the decisions on which features to develop and hence are the people who influence if and how product hypotheses are generated and validated.

To better understand the complex problem of *hypotheses generation*, a preliminary study will be conducted to find out how Product Managers currently make decisions on what product changes to develop. Amongst other objectives, the study sheds light on the roles that hypotheses, research, intuition, metrics, and experimentation play at the case study company. To shed light on these phenomena, the interview study asks the following overall research question: *'How do product management professionals make decisions about which product changes to develop in order to improve the software product?'*.

In order to reason if the developed DSR artifact has been ultimately successful in solving identified points, the author evaluates the artifact with a set of evaluation criteria. Evaluation subjects that are assessing the developed artifact are again the B2B company's Product Managers to close the loop.

### 3.2.3. Research Process

In general, the thesis research project greatly builds on the stages of Peffer et al.'s process framework [3], whereby following a Problem-Centered-Initiation. In order to generate in-depth knowledge about the problem space, a semi-structured interview study is conducted with the core product decision-makers at the case study company. Semi-structured interviews are amongst the rigorous social science research methods that are advised for DSR projects [57]. In addition, the thesis researcher follows DSR scholars' advice to analyze the academic knowledge base to integrate existing knowledge in order to create a more effective artifact [57]. Reviewing the literature also allows to better delineate the contributions of the thesis project to the overall understanding, i.e. to create transparency when and how the author builds on prior knowledge [2]. Finally, the embedding of the DSR project into the greater landscape of both DSR projects and practitioner projects is presented in this thesis. According to DSR scientists, this strengthens the significance and reusability of knowledge gained [2].

# 4. Explorative Research

In this chapter, the author presents the interview pre-study that was conducted with product management professionals of the case study company. This explorative research was carried out to better understand the environment of study with a clear focus on decision-making for product features. This decision-making includes which product changes are developed, how a feature's success is measured, to what degree product changes are translated into hypotheses as well as if and how hypotheses are validated. These are all areas worth investigating to understand better how product improvement hypotheses are generated. Additionally, the interview study can contribute a nuanced view of what potential pitfalls and constraints in the decision-making process are that must be considered in an effective artifact design.

## 4.1. Approach

The author conducted 5 qualitative semi-structured interviews with the 5 main people involved in product management at the German SaaS company. The problem exploration study's overall goal, which is split up into 4 concrete research questions later on, is to find out:

**Overall RQ:** *'How do product management professionals make decisions about which product changes to develop in order to improve the software product?'*

For an effective design, it is important to understand hypotheses generation's role in the decision-making process once new features are developed. As hypotheses include the feature as the treatment to improve a certain target metric, the research asks:

**RQ1:** '*What are the different steps in product management in the course of deciding on a concrete feature to develop?*'

To understand which concrete information as well as which abstract ideas a hypotheses generation tool may need to integrate, the research asks:

**RQ2:** '*Which concepts and artifacts is product management referring to, researching, or creating in the course of the decision-making process?*'

Technological support often builds on the execution of rules, yet intuition is described by scholars as probably predominant in hypotheses generation. Hence, the research asks:

**RQ3:** '*What is the role of rules, heuristics, and intuition in the process?*'

Finally, the research aims to build a system that logically fits into product decision-makers' everyday tool suite. As a hypotheses generation software needs to know the delineation to other tools and when to best integrate with third-party software, the interview study asks:

**RQ4:** '*Which tools are supporting product management in their daily work which play a role in decision-making about product improvements*?'

Similar to Johanssen et al. [59], the author condenses interview responses into insights. The interviews led to an invitation to further sight documents and tools that product management in this company is using.

The remainder of the chapter's explorative interview study is structured as follows. In the upcoming section, the research method as well as the design of the explorative study is illustrated. Following, the results are presented in a categorized fashion while also providing a condensation of results. Closing, the discussion section summarizes and interprets the results and derives implications for subsequent artifact design. Worth noting, limitations as well as overall knowledge contributions of the interview study are presented later in the course of discussing the overall thesis project in chapter 8.

### 4.1.1. Research Method

The information to answer the research questions has been collected in the course of a semi-structured interview study. Studies that apply semi-structured approaches have been found effective in domains that are characterized by complexity, diverse points of view, and the need for deeper-level insights. Concretely, semi-structured interviews allow for clarification of interviewees' answers and probing, i.e. asking non-planned follow-up questions [60]. Similar explorative studies in the field of software engineering also leverage this technique to allow for a great resolution on the domain of study and the collection of unforeseen knowledge [59]. This yields rich information about the environment and requires careful analysis to derive sound insights about problems to be addressed with subsequent design.

#### Design and Planning

The structured part of the semi-structured approach constitutes a set of pre-planned guardrail interview questions that have been derived from the overarching exploration research questions. Conditional sub-questions for potential follow-up probing have already been designed upfront. Otherwise, room for probing and clarification questions has been given. The interview setting has been pre-tested in a dry-run[1] with a Product Manager who has been working in a similar context in order to iterate on the interview design and guideline[2].

[1] cf. subsection A.4.1
[2] cf. section A.5

#### Data Collection

Due to the COVID-19 pandemic at the time of the pre-study, the interviews have been conducted via an online conferencing tool. The interviews were video recorded with the interviewees' permission and were transcribed in textual form. Because of the modest sample size, the author reports individual quotes or insights in a pseudonymized fashion (e.g. S1 for subject 1) without a clear mapping to a single person to allow for more open and honest answers. In addition to the interviews, the author has been granted access to investigate the tools and documents that have been mentioned by interviewees to further enrich qualitative findings.

### 4.1.2. Sample

The sampling of interviewees builds on core ideas from snowball sampling where referrals from previously studied individuals are taken into account for the detection of further interviewees [61]. As not all relevant stakeholders with respect to covering the information need have been known upfront, referrals helped to detect important roles in the population that needed to be included [62]. In the interview study, the role of the User Experience (UX) researcher has been mentioned multiple times by other interviewees to shed more light on the research perspective. This led to the belated inclusion of this subject into the interview study.

No further referrals outside of the group have been provided. This led to an overall scope of five interviews which were conducted in December 2021 with a target duration of 60 minutes each.

All interviewees were briefed prior to the research including a short outline a few days before the interview. At the interview's beginning, a short introduction to the study as well as the collection of biographical data was conducted. The subsequent interview part which was recorded totaled 268 minutes for all 5 interviewees (recording's avg: 53.2min, median: 56min). After the recordings, two follow-up questions were answered by a study participant in text form for further clarification on a certain topic.

### 4.1.3. Data Analysis

During the interview, field notes as well as remarkable quotes have been noted down. After the interviews, the author went through the recordings and transcriptions multiple times. Similar to Hoda et al.'s introduction to *Grounded Theory*'s application for Software Engineering [63], the author first applied open coding to extract key aspects from the interview transcripts and assigned relevant text snippets with a code - the preliminary label. Then, the labels are aligned with each other during the course of further sampling of more and more interviews, also referred to as the *Constant Comparison Method*. Continuing, core categories have been developed based on a full investigation of all interviews. Based on these categories, selective coding is applied which results in the final set of label groups, its associated labels, and hence its labeled text.

To allow for relevant condensation, the labeling has been informed by the study's

research questions. A subset of the transcriptions (analogous to a training set) has been also labeled by an external person independently. The external person's results have been compared with the author's labelings while highlighting general commonalities and differences. Following a bilateral discussion of the two labelers' approaches, the labeling procedure has been refined[3]. Afterward, the labeling was conducted by the author himself for the whole sample.

Similar to Johanssen et al. [59], the author condenses observations into insights that are derived from interview responses. Each paragraph in the following results section condenses learnings from one or a few similar labeling categories to allow for a clear mapping between labels and condensation. Similar to Schermann et al. [20], the overall condensation is enriched with quotes from the interviews.

## 4.2. Results

### 4.2.1. Descriptive Findings About Interviewees

The now presented biographical information about interviewees is a combination of their publicly accessible LinkedIn [64] profiles and information that has been given upfront the interview. The information retrieved from Linkedin was confirmed or enriched at the beginning of the interviews with the interviewee's cooperation.

#### Current Role and Total Experience

| | Current Role | Product Management | Software Engineering | UX Research | Business/ Consulting |
|---|---|---|---|---|---|
| S1 | Product Manager | 6.4y (2) | 1.3y (3) | | |
| S2 | Product Manager | 5.1y (2) | | | 0.6y (1) |
| S3 | Product Manager | 5.7y (5) | | | 4.0y (3) |
| S4 | Product Manager | 3.3y (2) | 2.3y (1) | | |
| S5 | UX Researcher | | | 0.3y (1) | 5.2y (2) |

Table 4.1.: Role-Specific Experience of Interviewees

*Part-time roles are factored in with 50%; different companies the subject worked at in the specific role are displayed in parentheses; only SaaS product development relevant experiences are considered*

Four out of five interviewees are Product Managers (cf. Table 4.1), i.e. they are making product improvement decisions and are managing the development team's backlog. One interviewee is a UX Researcher which is a role at this company that engages in deeper research and collaborates closely with Product Managers on which features to best improve or which opportunities to pursue. The UX Researcher usually applies a broader perspective on the whole product than each Product Manager. Product Managers mostly

[3] cf. subsection A.4.2

focus on dedicated parts of the product. All interviewees can be defined as product management professionals at this company.

On average, interviewees have 6.8 years of experience in the field of SaaS product development. Two out of the five Product Managers have a software engineering background and used to work as a developer before transitioning to a product management role. Further two employees have a strong business background, whereas it needs to be noted that they have been in business roles in the context of consulting or business analysis in the software industry.

When it comes to the rough tenure at the current company at the time of the pre-study, the interviewee's average tenure is approximately 2 years. Two interviewees (S3 and S4) have recently joined. One interviewee (S1) has been part of the company since it was founded while S2 and S5 have worked there since ca. 2.5 years.

#### Educational Background

| | Highest Education | Study Foci |
|---|---|---|
| S1 | Master's | Software Engineering (o.e.) |
| S2 | Master's | Product Management (o.e.), other Technical Studies |
| S3 | Master's | other Technical Studies |
| S4 | A-Level | Software Engineering (o.e.) |
| S5 | Master's | Product Management (o.e.) |

Table 4.2.: Educational Background of Interviewees

*Study programs were grouped together once they shared major commonalities; oe: short for or equivalent*

Four out of five interviewees received a Master's degree at a university while one interviewee conducted a technical apprenticeship after leaving school with A level (cf. Table 4.2). Two out of five interviewees have an educational background in software engineering while two others mainly studied in a broader product management field. In contrast, S3 studied in a deeper technical field and transitioned to software or product management later in her professional career.

### 4.2.2. Process Perspective

This section aggregates the insights that were mentioned while answering research question RQ1: '*What are the different steps in product management in the course of deciding on a concrete feature to develop?*'.

#### Different Feature Type, Different Process Execution

> 'It depends on if you introduce a completely new one or if you work on something that already exists.' (S4)

Interviewees emphasize the most important distinction in how the process looks like is the question regarding the type of product change. Is it about launching a new set of features or iterating on already existing ones? Having not launched yet, the new feature necessitates a different type of thorough research and generation of assumptions. Iterating on, i.e. aiming at improving an existing feature, follows a more lightweight process as there is already knowledge to some extent on the problem and how the current feature is solving it. Additional cases, like developing an entirely new product or providing an existing product with a redesign, have been mentioned as corner cases, but are encountered less frequently. In terms of frequency, interviewees report that in startup companies, the relative share of shipping larger product changes as opposed to slightly improving existing features is remarkably higher than at incumbent companies.

### Strategy and Product Planning

> 'First it starts with [...] the vision of where we want to go' (S2)

Interviewees report that the company starts by first strategically deciding where they see the primary focus of their software product. This decision has multiple inputs where internal and external considerations are fused. This clarity about where to operate comes from internal convictions on how the company sees the market developing and what might be important for its key users. Further, aspects like the business model come into play. These considerations are concretized in a product vision which guides the question: where does the product want to go? This is understood as a baseline decision line against which new product changes are meant to be assessed.

Proceeding the initial seed for a novel feature to develop, the fundamental question is whether the potential product change fits the product vision. It may be the case that the development of a feature sounds worthwhile, but it would not be profitable in monetary terms. Or, even though some feature ideas might be interesting, the company would face major competition that has already mature solutions. For the case of a profitable space and no remarkable superior competitor yet, potential features to be developed would be added to the product roadmap, i.e. a timeline perspective of which features are planned to be developed.

### User Problem-Driven Research for Entirely New Feature Areas

> 'Always get down to the why. [...]. Don't touch anything if you don't know why you're doing it.' (S4)

Informed by the product roadmap, a roughly outlined problem domain is the starting point for the development of a novel feature. First, product management is framing the challenge and is thinking about ways to learn more about the phenomenon. Product Managers are interested in the people using their software, and how they experience certain situations and connected problems in their real lives, i.e. outside of the software. To shed light on this, interviewees report that they conduct qualitative, semi-structured interviews

with users. These interviews should yield a better understanding of the underlying goals the people want to achieve, and current problems where they struggle. This then creates the spark for opportunities with respect to problem-solving, i.e. the new product change.

Iteratively, with more and more interviews, a collection of information is compiled and informally condensed into insights. Insights in this context are basically informed assumptions about the ground causes for certain problems. To provide some early robustness for these assumptions, they are sometimes informally checked during further interviews to either find more support for an assumption's validity or to question its validity. Apart from conducting interviews, visits to corporate customers' sites to talk to users or asking customer-facing colleagues for their views on potential user problems have been mentioned. Also, treating oneself as a user to reveal one's perceptions regarding associated problems and their ground causes have been named.

Interviewees report in this process step report that proactively researched information is dominating and quantitative data plays a subordinate role in these types of universal problems. However, as one interviewee suggested, he tries to drill into data in order to find signals in current data points. These signals can lead to a better understanding of the overall customer journey with the product. When the research is considered done, the findings from different streams are blended, condensed, and presented in a larger group consisting of multiple Product Managers.

#### Usage Friction Driven Research

> 'After our initial release [...] the first thing that we did was basically research. What is the current status? How do customers use it? What are their frictions? [...] What are new challenges?' (S4)

Not only novel features are developed, but also existing features are reworked. Interviewees report that the type and depth of research share major commonalities with research for a novel feature, but there are differences. The key reason why it differentiates is the now-existing quantitative data. Due to the fact that the feature has been already released, data has been generated from current product usage. Therefore, feature usage can be analyzed quantitatively. Further, qualitative customer feedback might also exist from the feedback users give to existing functionality either directly or via customer-facing employees. Simultaneously, Product Managers themselves are using the software's existing functionality extensively on their own to understand potential frictions better.

Interviews report they use primarily qualitative information to reason about improvement potential. Does the available information point to an obvious usage friction, Product Managers report that they do not invest the time in conducting interviews, but rather assume the usage friction and define it as the problem to be solved. If an interpretation is not that obvious, they report to conduct user interviews to to achieve more certainty on the problem.

### Defining the Feature Challenge

> 'Then there's problem formulations probably like '[x[4]]% of users are not part of a team and therefore cannot do anything on the platform'. So then the idea was, [...] let's make it easier to join a team.' (S2)

Based on an overall understanding of the problem, an informal assessment is undertaken which results in a target-actual comparison regarding how far the problem has already been solved by the existing state of the product. Interviewees could not name a clear heuristic or formula to follow. The more formalized part of this comparison is comparing the target metrics the feature was planned to reach with the current numbers to detect the need and magnitude of improvement potential.

Regardless of which research was conducted (either finding out about general user problems or current usage frictions) - different findings are fused together to concretize them into a challenge to be tackled. One interviewee especially highlighted that not every problem needs to be solved. So, the Product Manager in this phase needs to carefully prioritize what important problems are and which ones might not be worth the associated development efforts. In this stage, interviewees report it is also important to include Senior Leadership early on when these types of decisions are made.

### Turning Ideas Into Product Change Concepts

> 'The [...] Designer [...] ideated together with the Product Manager different approaches how this could become easier. [...] After very early ideas, developers came into the process.' (S2)

Based on the exact problem to tackle, the Product Manager is paired with the Product Designer to think of rough ideas on how exactly to solve the problem with a new product change (new feature or an overhaul of existing ones). Depending on the type of product change, Product Manager and Product Designer are developing variations of different solution approaches whereby they discuss and merge different approaches into a promising concept for a product change. The Product Designer is then transforming this concept into a tangible design. Or, if more concerned with a new user flow - the designer builds a prototype that is up for validation with internal or external stakeholders.

The more uncertain the match between problem and solution is, the more variations are produced and tested in this concept phase. For very simplistic improvements (e.g. a new wording or a smaller design update) where the clear improvement can be derived from customer feedback only, no further validation testing with users is conducted. For a sanity check, the solution concept (e.g. wireframe, UI design, clickable prototype) is then occasionally feedbacked by customer-facing colleagues who can give their opinion on the solution concept. Moreover, lead developers are already included in the discussion so they can give opinions regarding ease of implementation. Based on this, it gets clear for all parties involved what could be a Minimum Viable Feature (MVF) that brings both value

[4]Exact percentage anonymized by researcher

to the users and can be developed in a short amount of time. Based on actively insourcing external and internal feedback (e.g. with optionally user testings of prototypes), the solution concept is then iteratively improved until it reaches a certain maturity.

#### Handover to Software Development

> 'The formal scrum process kicking in and well then it got into sprints, developers picked it up [...] and then it got released.' (S2)

Based on the product change concept and the discussion with developers on how to best develop the feature, the Product Manager is defining the concept in the form of user stories. The user stories are then defined in a very detailed manner for the developers so they can construct the software in such a way that it solves the problems exactly as defined. To allow for this accuracy, the Product Manager is not only defining the feature per sé but also an extensive list of edge cases and expected behavior.

Once the Product Manager has handed over the feature concept to the developers, software development starts. The work items for developers are then part of an agile backlog and the developers are executing the work items in Sprints. While developers are working on turning the product change into reality, the Product Manager is preparing the rollout already from her end. One interviewee highlighted the need to prepare rollout accompanying content for internal and external usage. This means content explaining new functionality, what has changed in the software, and why it has. At this point, there has been no mention of adding anything non-feature related, like setting up experimentation like implementing feature flags or choosing group assignment algorithms.

#### Post-Launch Analysis Leads To a Continuation Decision

> 'How well is it adopted? How did users like the release? We want to compare this post-launch case with our assumptions before and gain some learnings.' (S4)

> 'So what other people might call experiments or launching A/B tests [...], something we [...] don't regularly do.' (S2)

Interviewees report that they monitor feature rollout and its associated metrics over time. After a rough time frame (ca. one month), the Product Manager evaluates feature engagement in terms of quantitative data and conducts a target-actual analysis. In the course of this, she compares the target metrics they ideally have set for a successful feature rollout and looks at the metrics and how they have developed by this time in a pre-post comparison manner. This is paired with overall impressions of both, users' feedback and the overall intuitive impression of the Product Manager on how successful the rollout has been. At this stage, the Product Manager is using certain symptoms she derives from looking at numbers (e.g. for bad engagement with a feature) to ideally find out ground causes by actively asking users about their impressions of a feature. Taking all together, this leads to a decision if the feature should be further iterated, left like it

is for now, or rolled back. Based on the decision to continue, this may lead to further research to be conducted for a next iteration.

#### Roles Mentioned Within the Process

During the lifecycle of a product change's development, not only Product Managers and Software Developers participate in certain tasks. Interviewees mentioned besides their own roles (Product Manager, UX Researcher) also the role of the Product Designer who plays a part in developing different solution ideas and visual prototypes. Three further internal roles come into play when quasi-validating assumptions or getting new insights: the role of the CEO and Senior Leadership, the Sales Manager, and the customer-facing Customer Success Managers. All three groups are in closer contact with customers but also have strategic considerations in mind and thus serve as internal stakeholders for the process. Outside of the firm's four walls, the role of the customer as well as the role of the user has been mentioned. While a user is an individual who uses the software, the company is the entity that purchases user licenses for hundreds or thousands of its employees, the users. When looking at the outside world, the abstract role of the competitor must be mentioned. This role subsumes all relevant software companies that aim at solving similar problems for the same or similar customers. Drawing conclusions from competitors' products and features might inform feature development as well.

### 4.2.3. Concepts and Artifacts

In this chapter, the author investigates aspects that aim to answer the research question RQ2: '*Which concepts and artifacts is product management referring to, researching, or creating in the course of the decision-making process?*' With concepts the author means core ideas that Product Managers refer to while artifacts are when those concepts are externalized and exist as e.g. a document. Because of the different flavors and flight levels of concepts, they are separated into different sections.

#### Core Mental Models Framing the Context of Product Management

> 'You know like the search space metaphor. [...] Combinatorially, if you would research everything you will have like the combinatorial explosion [...]. So [...], you limit the search field.' (S3)

Interviewees mention several core narratives to make sense of the complex working world they operate in. While Product Managers' mental model is manifold, four core parts of their mental model shine out from observations.

First, they perceive to be working as Product Managers in two interconnected abstract spaces. In multiple interviews, participants mentioned *problem space* and *solution space*, or alternatively two diamonds[5]. Moreover, they also describe each space as a *search*

[5]The author assumes interviewees meant double diamond, a framework developed by the British Design Council since 2004 [65] which describes activities in the problem and solution space

*space*. For a given problem in the problem space (that contains an infinite number of problems), there exists an infinite amount of possible solutions in the solution space. Product management is thus perceived as an activity of limiting the scope of each search space to manageable sizes and then deciding on clear problems to tackle and clear solutions to develop.

Second, interviewees strongly refer to degrees of *certainty* or *uncertainty* which frames their work in the respective search spaces. One interviewee explicitly described product management in situations of little a priori knowledge as an act of 'bets and gambling' (S3). In other cases where information is abundant and there's little risk of failure, Product Managers' work is seen as 'ship it, relax' (S3). Following from this, research, experience, and hence access to information is the lever that can decrease uncertainty.

Third and tightly connected, in order to reduce uncertainty, Product Managers often refer to *ground causes* for certain superficial problem symptoms, also referred to as *the why*. Gaining deeper knowledge is the prerequisite to understanding the composition of the search spaces better and selecting ideal problems to solve.

Fourth, interviewees mention the special role of a product being in the *startup maturity* stage. This differentiation connects to the previous points in the sense that product management in a startup stage means being confronted with a setting where features are often not mature enough and the product might not have stabilized its direction. Similarly, the share of big feature launches is higher than in more mature products where tiny optimizations of existing functionality are the core activity. While this indicates that probably higher optimization potentials for product changes exist in immature products, product management in startups does not see themselves primarily as engaging in optimization.

Fifth, interviewees mention general working modes as part of their mental model which fit largely into the categories of *agile* and *customer-centricity*. First, interviewees mention the general statement that striving to develop solutions in an incremental and fast manner is seen as overly positive in their mental model. Second, slicing product teams not necessarily in functional domains but according to customers' problem fields and user journeys is seen as a pattern that is worth pursuing.

#### The Product Manager's Self-Perception

> '[The] PM job is very context-dependent.' (S3)

In the previous paragraph, the general mental model has been outlined. But, interviewees not only rely on certain narratives in their context, but they also make sense of their role in this context. Interviewees report consistently about different natures of the Product Manager, i.e. a single person wearing different hats.

Interviewees perceive the Product Manager as a *problem researcher* and *opportunity selector*. In terms of research, this means that the Product Manager is investigating potential problems users have, understanding them deeply, and finding out how worthwhile it would be to solve certain problems. Moreover, interviewees refer to a *strategist* aspect of the Product Manager, i.e. she is reasoning about which market the company is operating

in, and which problems are worth tackling with a new feature. Also, Product Managers see their role as *product owner* and guide software development with the help of crisp feature requirements and feedbacking / testing the work along agile development processes such as Kanban or Scrum.

Apart from wearing different hats, interviewees see themselves as highly individualized actors operating as Product Managers. Interviewees report they may draw their inspiration from prominent voices in the field of product management (e.g. Marty Cagan [66]), from agile frameworks, or idealized notions like *the data-driven, rational Product Manager*. However, interviewees mention they always bring their specific individualism to these idealized views on product management as their work is considered to be domain-specific. The specific context of the company makes taking on a blue-print complicated and always necessitates some twist and careful application of best practices to get the job done.

A large topic that interviewees report is that while leveraging intuition, a Product Manager's intellectual antagonist is her own bias or the bias introduced by listening to others. The Product Manager must be aware of various possibilities of biases and always think about how she can reduce them. As some examples, mitigating confirmation bias for own feature ideas, biases stemming from low or too early reviewed qualitative samples, or following Senior Leadership's ideas have been mentioned by interviewees.

### Search Space Narrowing Artifacts

> 'We also see hundreds of other problems where we could potentially add value, but it may either not be our space or our belief.' (S1)

As illustrated above, Product Managers are (ideally) solving worthwhile problems that reside in the problem space with valuable solutions from the solution space. But putting boundaries in both spaces is important otherwise Product Managers cannot move forward. To rationalize search space restrictions, product management relies on shared artifacts and concepts.

Interviewees report that the company has agreed upon certain *key personas*. These are qualitative descriptions of users, the contexts they are living in, and which relevant problems they likely want to have solved. Interviewees report that this artifact helps in knowing on which people to focus on during research and development, thus shrinking the problem space.

For problems, different concepts have been mentioned although they share major commonalities. Interviewees almost consistently referred to universal problems as *jobs to be done* (or a synonym: *user need*). A job to be done is considered to be an aggregate information piece that frames a key persona's goals she wants to achieve in a certain domain. Jobs to be done are grounded in insights of a mostly qualitative kind and exist as a shared artifact. On the other hand, *usage frictions* (or the synonyms pain points, blockers) are as well problems but they are rather solution-bound. Usage frictions would describe e.g. problems to discover, understand, or use a feature. According to interviewees, problems are also concretized in problem formulations.

While human beings might be the starting point of a customer-centric restriction to the search space, other strategic artifacts come into play as well. Interviewees report diverse higher-level strategic artifacts such as the *company's vision*, *mission*, *value propositions*, *business model*, positioning in the *tech landscape*, and *shared beliefs* about the domain. These are perceived as informal guiding ideas to inform day-to-day product decisions, rather than clear artifacts that have clear implications.

#### Qualitative and Quantitative Information Artifacts

> 'Then we [...] kind of went through hundreds of requests.' (S2)

> 'I would say looking at adoption rate of features, right?' (S1)

To make product decisions, interviewees mention both qualitative and quantitative information which is reviewed. An extensively referred to source was qualitative observations. Qualitative information is either inbound approaching the company or is actively searched for outbound (e.g. conducting interviews and transcribing them). To give a better understanding, a quick insight is given below.

When products and their features are being used by users, some of these users are proactively expressing their feedback. This can take the form of concrete *feature requests* (missing functionality), *user experience feedback*, *support tickets*, or even *product reviews* - although the latter seems to not play a larger role. These information pieces are insourced via different channels. These information artifacts are collected centrally in a structured format with metadata on the author (user), company, and content of the feedback/request. Also, when conducting proactive user interviews, these interviews get recorded and transcribed. These *interview transcriptions* get optionally content coded by Product Managers or the UX Researcher to condense *interview quotes*. These interview quotes get then optionally clustered. Another way to receive insights was informally communicated information to customer-facing employees during Sales pitches or calls. This vague information is concretized by customer-facing employees and insourced in a similar format than illustrated above. Moreover, a rare mention was that qualitative feature usage once information on text inputs is available is used to learn more about how users are likely using a feature.

On the quantitative side, interviewees report that they rely on data for basically two use cases: evaluating the success of their feature rollouts by comparing *actual product usage* with expected usage (reality vs. expectation). On the other hand, they rely on quantitative data to learn about usage and to explore *potential improvement potentials* with an open mind. In the course of this, interviewees report they are e.g. conducting specific analyses like *funnel analysis*. Further, *custom explorative data analyses* are conducted to learn more about usage. For the case of evaluating the success of a feature, mostly *target-actual analyses* are conducted and evaluated to reason about the success of a feature rollout. Explicit statistical analyses or A/B testing results have not been mentioned by the interviewees. Taken together, the information artifacts are both data points on a detailed level, as well as data analyses and their results. Often the analyses

do not have a clear goal in mind but are of an explorative nature - and sometimes the Product Manager's individual intuition paired with the right analysis is fusing into an epiphany moment.

### Tacit Knowledge and Experience

> 'I'm building my mental library of user interfaces, ideas I can steal from.' (S3)

One source of tacit information[6] is reported by interviewees that they rely on external (to the company) information for enriching internally available information. This may be *domain-specific knowledge* from product management blogs they have read in the past. Further, *observational experiences* from actively trying out products of close competitors or their own software product have been mentioned. In addition, an interviewee mentions playing around with analogous products, i.e. companies operating in very different markets, but offering a relevant experience. While this could theoretically be externalized, there were no larger signs of transcribing this knowledge. The richest but also most complex set of non-externalized knowledge is probably situated in the year-long experience of Product Managers in their engagement with products and development of software products. This tacit knowledge is sometimes quasi-queried to relate new insights to existing intuitive knowledge one has gained over the years.

### Condensing and Ordering Artifacts

> 'So that's a collection of anything a [...] customer or user mentioned.' (S2)

> 'There is a gap of some kind and then there is a repetition of some kind [...]. Actually, I look for patterns.' (S3)

Information is best usable if it is externalized and accessible. Externalizing information was mentioned throughout the interviews in different forms. Regardless of which artifact is reviewed, Product Managers and UX Researchers alike try to condense this diverse set of information into *insights* (or the synonym *learnings*). It describes an abstract term when inducing knowledge that comes from the condensation of repetitive or surprising information. Synonyms for recurring information or gaps have been mentioned in this course as *signals* or *patterns*. Also, the misuse of features has been described in this context. Insights and patterns were mentioned both in active qualitative research and once analyzing real usage data after rollout.

One of the interviewees highlighted the importance of centralizing information artifacts. The interviewee states that information artifacts are often dispersed in many places. Product management started to collect and structure information in a form similar to a knowledge management system. Apart from Product Managers and UX Researchers, employees from different departments can contribute *raw information* to this knowledge base. Product Managers focus on putting structure into the collection of raw information

[6] Tacit knowledge means knowledge that is not externalized, i.e. not documented in artifacts

artifacts in order to give them meaning, e.g. to which features a certain information artifact belongs. Once this information is externalized, collected, and given relevant context, it can be used in the long run by Product Managers to build a collective customer intelligence. In fact, multiple Product Managers reported that they are browsing this knowledge base for exploration and looking customer views up once they are in the process of feature conceptualization.

#### Solution Artifacts and Their Validation Concepts

> 'This is a problem you want to solve for the user and then you set up a hypothesis [...]. And from that hypothesis, the prototype is derived.' (S1)

> 'You make educated guesses about something and then you collect data to ideally confirm or disprove it.' (S3)

Before proceeding to feature development, interviewees report they build a lot of conceptual designs. This spans the early stages of an idea from textual descriptions of an *idea* to *wireframes* (or mock-ups) to *high-fidelity clickable UI prototypes* with enhanced user flows. Product Managers refer to artifacts in different maturities often with the abstract term *concept*. Once the solution is developed in code, Product Managers refer to a product change mostly as a *feature*. However again, interviewees also use more high-level terms for the solution (*product*) as well as more fine-grained terms for referring to a solution (*functionality*). When the feature itself is a first rollout with just rudimentary functionality, multiple interviewees referred to the term Minimum Viable Product (MVP). Features are also named in terms of how they interact with other components in the software - indicating some collection of functionality.

To validate if solutions are actually solving the problems identified, terms like *assumption* and *hypotheses* have been mentioned. Interviewees consistently used the terms interchangeably and described with them the rather generic concept of an *educated guess*. No interviewee has mentioned a hypothesis in a more formalized fashion that would make it apparent how it would be validated exactly. While there was no clear formalization of a hypothesis there have been mentions of operationalizations that are somewhat drawn to reason about the success of a feature. These have been mostly described as *success metrics* - and more explicitly meta metrics concepts like *retention* and *adoption*. Ingrained in the operationalization was a kind of expected delta between the current value of the metric and the desired value of the metric.

Regarding (quasi-)testing the validity of assumptions, some interviewees reported that some sort of data is collected and *post-launch analysis* is run. Leveraging randomized controlled A/B tests or similar experiments has not been mentioned. As a more explicit test regime, only guided *user experience tests* with few selected participants have been mentioned. The are no mentions of formalizations of what exactly a test or an experiment is, starting with the hypothesis. Besides that, *internal sanity checks* (non-formalized validations) with other Product Managers or colleagues to challenge their own assumptions or feature ideas have been mentioned. Overall, interviewees rather mention hypothesizing and validation in vague terms.

#### Ambiguity of Concepts and Their Relations

> 'You also need some kind of concept to get to a prototype. But this is not the concept you will bring to your devs or bring into a company.' (S4)

A lack of precision, definitions, and alignment in the use of concepts and artifacts stands out in the answers of all interviewees. All interviewees report that they do not draw a clear distinction between what they call a hypothesis or an assumption while using both terms regularly. Further, when talking about problems to solve, interviewees use synonyms such as pains, blockers, or frictions - yet it is not paramount that they are necessarily talking about the same things. A further term such as a concept seems to be both an abstraction (the general abstract term behind the idea, wireframe, or design) and a concrete entity (such as a concept in the form of a very detailed development briefing). Terms such as concepts, assumptions, decisions, and validations are used in multiple phases of the product development cycle. And, following this, the concrete context needs to be taken care of to understand its context-dependent meaning. When asking follow-up questions, interviewees could see that the terms tend to be highly overlapping, on different levels of abstraction, and realized that their role in the decision-making process might be less straightforward than initially expected.

### 4.2.4. Rules, Heuristics, and Intuition

In this chapter, the author investigates aspects that aim to answer the research question RQ3: *'What is the role of rules, heuristics, and intuition in the process?'*

#### Implicit Rules and Suggested Heuristics Dominate

> 'Maybe there are rules with many exceptions [...]. No doubt that there are hidden rules [...]. It's a very complex system.' (S4)

Interviewees have been confronted with questions regarding when they use clear and explicit rules throughout the process.

Only a few responses were made to this point and often mentionings of clear rules have been softened once interviewees outlined them in greater detail. In principle, an explicitly followed practice seems to be that features should always be accompanied by sufficient data-tracking capabilities so that subsequent quantitative analysis is possible. On an implementation level, further explicit rules have been mentioned in the compliance of filling certain templates when it comes to documenting decisions, or to document the feature's concept thoroughly when handing over to developers.

Interviewees report mostly implicit rules and heuristics they follow while acknowledging it might be hard to make such rules explicit. There is anecdotal evidence of areas where implicit rules are most likely shared by majorities of Product Managers, e.g. when it comes to connecting assumptions about feature releases with mismatches between expectation and reality when looking at a rollout's success metrics. Some collaboratively followed heuristics might also exist in the way interviews are conducted and how information is

best interpreted. A noteworthy heuristic when looking at data was that triangulation or cross-checking (i.e. investigating a phenomenon with multiple research methods, e.g. quantitative and qualitative) has been seen as a good practice to follow. While presenting mostly implicit rules, interviewees report that they have suggestions on rules to follow. However, while suggested heuristics may be beneficial for guidance, interviewees mention that such rules might never be entirely followed. Corner cases have been mentioned regarding why it is not paramount how exactly to define rules that are able to grasp the complexity of the process.

### Creativity and Experience

> 'Creatively [...] you sit down with the designer and you design the best solution you can [...]. It's not deduction.' (S3)

> 'It takes some muscle and also [..] having launched like 10 different features. Get a feeling for what success will be like.' (S1)

During the process, there are several instances where interviewees refer to creativity in contrast to heuristics or rules. Prominently, the unstructured process of brainstorming ideas to solve a certain problem or which design to take have been mentioned. Further, interviewees name other situations such as asking the right questions during interviews or factoring in abstract strategic business considerations. These activities are hard to frame in logical terms and are attributed to the intuition and creativity of the Product Manager, Product Designer, or UX Researcher respectively. However, it seems that creativity is always applied with an overarching goal in mind, may it be selecting worthwhile opportunities or brainstorming valuable product changes.

One interviewee reports that she *feels* something intuitively that a feature launch will be promising although the data might not tell it yet - as special dynamics like the virality of a feature come into play. Another interviewee who recently joined mentioned that a great challenge for a newcomer is to set good motivating success metrics for feature releases. Interviewees also report they leverage their individual experience of the past to reason based on it or to relate new research information to it. Product Managers who have been at the company for a long time highlighted their reliance on previous experience and knowledge.

### Practices While Inducing Patterns and Insights

> '10 people talking about the same job to be done in a row, then this is usually something where you can say: okay this is actually wow.' (S1)

Interviews indicate that a lot of heuristics relate to inducing assumptions from existing or researched information. Interviewees seem to think a lot about how to make their induction from qualitative or quantitative research more trustworthy. While no clear and enforced guidelines for sample sizes, compositions thereof, or significance of quantitative

analyses have been mentioned, there are indications of implicit actions shared by Product Managers.

First, Product Managers seem to be aware that minimum sample sizes for interviewees and fair compositions of the test sample are worthwhile and try to factor these considerations in to some degree. Second, implicit practices have been mentioned towards in which way to ideally analyze quantitative data in an explorative manner (e.g. cross-checking with other sources). Third, practices to search for repetitive occurrences in interview quotes (which is reminiscent of theoretical saturation), anomalies (or gaps) could be identified. Fourth, to not base pattern or insight induction on one source only, they implicitly search for qualitative and quantitative sources to see if the same patterns emerge with other means (which is reminding of triangulation). Fifth, as some kind of sanity accompanying heuristic, actively researched information or information to review is contrasted with previous knowledge. This might be general experience or company-specific experience.

### Deduction of Assumptions

> 'I would say an assumption is always based on some sort of insight.' (S4)

Interviewees report that there is a logical path to follow to end up with an assumption for a product change. The assumption is ideally backed by insights and ground causes why a certain product change is probably a good idea. This heuristic has been named by multiple interviewees and is also semi-enforced by decision-making templates. However several interviewees report that the company made proactive decisions that might not be supported by its customers yet (so not based on insights) as they have a strong conviction that it is the right strategic move.

Besides making assumptions about which feature to build, also making predictions about the expected positive effect of product changes has been mentioned. Most tangible assumptions have been named in the direction of which features to build and how, but predictions stay vague and not testable.

### Bias Mitigation

> 'I try to avoid to follow the loudest [or] the most powerful voice usually because it's kind of like a bad style of management.' (S3)

Several interviewees mentioned the role of biases in the course of deciding on a product change to develop. When one cannot always rely on a magnitude of data, it is a risk for product management to fall for certain biases. Interestingly, mentions of confirmation bias have been quite high throughout the interviews. A Product Manager ideally asks open questions and challenges her own assumptions to not fall in love with her own ideas, assumptions, and decisions. Especially when potential decisions are pushed by employees higher up the rank or by colleagues from the sales department, it needs careful research to not fall into confirmation bias. The same goes for loud customer voices, i.e. users who communicate their needs more proactively and dominantly. Apart from being bias-aware,

they debate their assumptions and induced insights with other Product Managers to get an outsider's perspective. Further, Product Managers are talking to customer-facing employees outside of the product teams who have deeper knowledge about customer claims. If they deem something as being reasonable, Product Managers perceive this as further proof for an assumption or decision to be probably robust.

#### Uncertainty as Moderating Factor

> 'Sometimes it's obvious what needs to be done and [..] we do like one prototype. It totally fits what we want to do and we just go with it.' (S3)

> 'It's often the case that there is some disagreement and this can then be often solved by saying: okay, let's do the first iteration and then continue to learn.' (S2)

Interviewees report that not all product changes are associated with the same degree of uncertainty. When it is quite obvious what needs to be done, the Product Manager is often just proposing together with the Product Designer the one obvious solution for a problem. In other cases, when the problem is generic, one interviewee said she is relying on best practice UI patterns to design the solution without any greater validation. If the feature is more complex and it is not as obvious how to solve the problem, the Product Manager and Product Designer create multiple variations of features for validation. Another mentioned strategy was to do more research and seek existing sources to triangulate and decrease uncertainty. To further reduce the associated risk of developing a wrong feature, the feature iteration's scope might be defined as smaller and released in order to learn more about how to improve it.

### 4.2.5. Product Management Tools

In this chapter, the author investigates aspects that aim to answer research question RQ4: *'Which tools are supporting product management in their daily work which play a role in decision-making about product improvements?'*

Interviewees mention a tool suite consisting of ten core tools that are targeted at the product management process. In the portrayal here, the author focuses on dedicated tools for product management, whilst excluding generic productivity tools (e.g. presentation software or messaging apps). Besides generic tools, there are five core types of software tools. While some tools focus on research and analysis, other tools focus on conceptualization, documentation, or testing. The author now briefly illustrates the tools and their usage.

#### Qualitative Research Tools

The company uses two tools that support its qualitative research endeavors. On the one hand, *condens* [67] is a software that allows Product Managers and UX Researchers to conduct user research projects and associated interviews. The company uses the tool for

collecting information about each interview and the automatic transcriptions of research interview recordings. Further, the Project Manager or UX Researcher are superficially labeling the transcribed text and are conducting a synthesis of those labeled text snippets into concrete research insights.[7]

On the other hand, *productboard* [68] is a tool specifically targeted at the Product Manager. It serves as the central hub where qualitative insights and planned feature releases are merged. With productboard's integrations, diverse qualitative feedback like feature requests, feedback emails or messages or manually provided memos from internal colleagues flow into the system. These feedbacks are then also labeled with either broader jobs to be done or the features the feedback belongs to.

### Quantitative Research Tools

Besides tools to support qualitative research, product management uses two different applications to explore quantitative data. First, the product analytics tool *pendo* [69] is used to track and analyze web analytics information from the software product's users. The tool stores data on click or hover events, in-app survey responses, as well as pages visited. The software provides means to analyze this data, e.g. in the form of funnels or overviews for monthly page clicks.

Second, the company built its own analytics report based on data that backend teams store in the relational production database. Based on the data that is stored, aggregate metrics are calculated in a data warehouse and visualized in *tableau* [70]. Tableau's visualization serves more the purpose of keeping track of the company's main product metrics (like monthly active users) over time than measuring individual feature rollouts.

### Ideation and Conceptualization Tools

Because the maturity of a solution differs heavily from the initial idea to a refined prototype, Product Managers use two distinct tools in the process. *Miro* [71] is used mostly when brainstorming solutions in remote meetings with multiple attendees or when presenting concepts still in their infancy. Further, the Product Managers are using *figma* [72] to view, test, and feedback the designs and prototypes mostly built by the Product Designer.

### User Testing Tool

While the company does not use clear experimentation frameworks for its feature rollouts, it uses a prototype testing tool called *maze* [73] to test its clickable prototypes early on to get an early validation if their feature concepts are going in the right direction. The tool collects primarily qualitative feedback from the feedbackers who can be internal colleagues, or dedicated test users from the user base.

[7] The document sighting showed that labeling or synthesis is not often the case, likely due to time constraints.

#### Development Workflow Tool

Product Managers use both atlassian's *jira* [74] and *confluence* [75]. Jira is used for defining and planning tickets (with the defined concept to develop) for the engineering teams. Further, confluence is used as a general tool to document all noteworthy information such as important decisions regarding product conceptualization. It is mostly the final place where documentation ends up while pieces might be upfront in other less formal tools such as the canvas tool miro. Also, the company uses the software *workpath* [76] to break down strategic considerations into tangible quarterly goals that are influencing mid-term product management priorities.

### 4.2.6. Reflection on Status-Quo

So far, the paper has provided insights into the four research questions posed. However, relevant side findings have also emerged and are now presented by the author.

#### Preference for Speed and Iterations

> 'If you want to be quick, you want to ship something quickly and iterate quickly. And learn as fast as possible.' (S1)

Interviewees repeatedly point out that speed is a very high priority in product development. Ideally, a feature idea should be brought to the user as quickly as possible. Having launched to real users makes data tracking possible and allows for subsequent analysis if it has been adopted quite fast. Furthermore, not only speed per se was considered important, but also the frequency of further product improvements. A widely mentioned bad practice was one case where a feature has been developed where no further product change to improve the experience has been developed to date.

#### Wish for Thorough Research and Robust Hypotheses

> 'There could always be more research [...] pre-launch, post-launch.' (S3)

> 'Even more important than to do proper research is to look at the data [...] this is like actual feedback right?' (S1)

Besides speed, Product Managers wish to build their work on solid ground, namely robust assumptions derived from thorough research. Investing enough time for research, analysis, and validating ideas before time-consuming development has been mentioned often. Besides these points, the wish to analyze the data more thoroughly and make hypotheses more robust by leveraging more quantitative data has been mentioned a lot. Overall quantitative usage data has been seen as the highest quality of information to both induce insights or validate assumptions.

### Wish for Standardization and Accessibility

> 'We're not good with like writing down the facts we've collected. So it's very opinionated. [...] But ideally, it would be cool to have some kind of like facts and research' (S3)

> 'It's an ongoing kind of bringing order to chaos.' (S2)

One interviewee highlighted the ongoing efforts of streamlining the research findings and using their tool landscape properly. According to Product Managers, only information that is externalized, collected, and organized can be effectively used while deriving insights. Several interviewees report that they face challenges in easily navigating and accessing the relevant data that they are interested in. As the company uses different tools and data sources to calculate similar (yet sometimes deviating) metrics, this demonstrates a major challenge.

### Time and Resource Constraints

> 'On like a meta-level, I would say that the biggest challenge here oftentimes [is] being quick and shipping something [...] and how much time you spend to really understand the problem.' (S1)

> 'At startups [...] products tend to be unpolished [...]. Because resources are limited, like you ship whatever, which kind of solves the problem.' (S3)

Interviewees mention that their product development time and resources are quite limited due to resource constraints in a startup setting. Generally speaking, lack of time was putting the constraint that one needs to decide how much capacity to spend on thorough research and deriving stable assumptions to cutting the time and moving earlier to development. Interviewees mentioned that research and thorough analysis are often cut to proceed further with more opinionated product assumptions. However, the costly drawback of rushing too fast in development has also been mentioned with the cost of increasing technical debt or making the wrong decisions.

### Cognitive Constraints and Benefits of Humans in Product Management

> 'I try to limit the flow [...]. [If tools] notify you all the time [...], you just lose.' (S3)

Interviewees also report on the human aspects of being a Product Manager. Besides reporting on biases of humans, they also report on context switches and connected attention problems. Some interviewees report that it is hard for them to manage daily business and product discovery which builds on in-depth engagement with a certain topic. Also, a human was described to be constrained to only focus on a few things at the same time. Also, humans as being part of the overall organizational power structure, which

gives them a blur of their perceptions, have been mentioned. However, the human in the equation can also bring some context to product management which would rather be inaccessible by technology. Beliefs and convictions on what users really need have been mentioned especially by one interviewee with a long tenure at the company. These are concepts that one would especially attribute to thinking and feeling human beings.

#### Complexity of Inference from Data Analysis

> 'You have attribution of different actions that you do at the same time, the same product, which can lead to an increase in engagement. And it's difficult to tell if it was feature A, B, or C.' (S2)

One Product Manager in particular, who places a lot of emphasis on data analysis, highlighted the complexity of deriving meaningful insights from data analysis. The definition of what would be good success metric and how are metrics related to each other seems to be of a bigger complexity. For Product Managers, it seems to be complicated to translate a product change's impact on overall ongoing and sustained user engagement as they seem to miss an overall framework or process that is connecting the dots.

## 4.3. Discussion

In this section, the author summarizes and discusses the learnings from the interview study that shed light on the decision-making of product management professionals at a B2B SaaS startup. First, results are summarized and second, the generated design knowledge for subsequent artifact design is presented.

### 4.3.1. Condensation of Results

The interview study has yielded rich insights to answer the main exploration research question: *'How do product management professionals make decisions about which product changes to develop in order to improve the software product'*? The main themes in relation to the research question are summarized below.

Product Managers at this company are working in a fundamentally agile product development environment and see themselves as problem solvers. Product Managers' main considerations are to understand the user as best as possible and at the same time to launch features fast and to learn from failures and successes. To make sense of this, Product Managers rely on an abstract mental model of two search spaces where they search for worthwhile problems to solve and powerful solutions to address these problems. In general, there are two types of problems, the ones that are fundamental to users' lives and the ones that are introduced by the software product. Both require applying a slightly different decision-making process as a Product Manager. Besides inducing insights from data and qualitative information, strategic considerations of the company and the Product Manager's intuition were seen as the main points which problem areas are worth tackling. In order to get a better picture of how exactly problems look like, Product

Managers are undertaking both qualitative research and quantitative data analysis - which serve different yet complementary purposes. To make assumptions about which product changes to develop more robust, Product Managers occasionally use validation techniques before rollout to improve the reliability of assumptions. Finally, a set of software tools targeted at different relevant domains such as research, data analysis, insights collection, and documentation is used at the company.

In general, the company is found to make product decisions rather in an *induction-heavy and deduction-light* paradigm. This means that Product Managers are actively researching information to create theories about who their users are and which problems they probably have. They do so by inducing their theories about users by reviewing diverse pieces of information such as customer feedback, feature requests, and interviews and blending all this with their observations, experience, and data analysis. However, when it comes to deduction (cf. [77]), generating a hypothesis and testing it to generate knowledge is seldom done. Most interviewees report that they are continuously making many assumptions, but state that validating these assumptions is rarely done in a high-quality fashion. In support of this, gold standard experimentation such as randomized controlled experiments (e.g. A/B tests) seems to not be systematically applied. A practice that comes closer to A/B tests, while potentially having many flaws, is what they call post-launch analysis. That is a practice where Product Managers review post-rollout usage data in an explorative manner to explore if some metrics or graphs have moved since the rollout of a product change. While perceiving data as the most honest feedback and feeling the urge to become more data-driven, interviewees report that by working in the current way, they find problems in attributing rolled-out functionality to changes in important metrics. To conclude, the product organization seems to be quite lightweight when it comes to defining hypotheses with a clear pre-rollout expectation and connection to experimentation.

When it comes to the relevance of formalized rules and heuristics versus intuition, creativity, and experience, many Product Managers describe large areas of their work as relying on the latter. When it comes to creativity and intuition, two areas in which they are applied are most dominant. First, as search spaces are described to be infinite and time is limited, it needs a confident going forward with the help of intuition. The skill of the Product Manager to assess a potential product improvement regarding strategic fit is likely demanding a lot of diverse knowledge. Second, the combination of intuition, creativity, and experience is leveraged during brainstorming. Coming up with different product changes in order to improve the product was a common case where rules and heuristics were seen as hard to apply. In general, formalized processes and rules were only occasionally found and informal practices dominate. To summarize, the company is delegating product decisions mainly to the individual Product Managers without strong processes or rules in place. Product Managers then tend to strongly leverage their complex mix of intuition, experience, and creativity. Being aware of potentially falling for their own biases, Product Managers present their assumptions to each other and challenge them if necessary. Once interviewees demonstrate their perspective on intuition, it is not necessarily seen as a flaw in the product management process. However, it is seen as a necessary talent to move forward in the infinite search space.

One core conflict that seems to have an influence on the potential application of more scientific rigor can be identified. As resources are constrained in the startup context, applying more scientific rigor was found to come potentially at the cost of moving slower with rolling out and iterating features. Interviewees are aware that they operate within these constraints and wish to have both higher scientific rigor and delivery speed. Hence, a core skill of a Product Manager is to carefully assess each situation when a new product change is conceptualized on which side the certainty-speed trade-off is leaning. There might be cases where a product change is a seemingly obvious improvement. Applying extensive testing or research to come up with a good improvement hypothesis might be too much on-top work and would slow down delivery speed. For other cases, there is more uncertainty, so applying more scientific rigor might be preferred. It has to be noted that this argumentation was mostly related to engaging more in pre-rollout research and validation of prototypes. Yet, there are few reasons not to apply this trade-off to post-rollout experimentation as well. In fact, one could assume that the missing use of higher quality experimentation might be partly explained by this certainty-speed trade-off which is having a preference towards speed at this company. To support this notion, informally checking posterior numbers without setting up an experiment does not significantly slow down delivery speed.

### 4.3.2. Generated Design Knowledge

The investigation of the application environment yields not only knowledge about the domain but also generates knowledge about worthwhile design options to pursue. In particular, the deep examination of the product management process, the role of creativity, rules, intuition, induction, and deduction allows the researcher to create an effective collaborative design between man and machine in the context of a DSS. Furthermore, parts of the hurdles of CEx adoption could be identified fairly well, i.e. the cause can already be found to a relevant extent in the informal and vague hypotheses generation which does not lead to statistically testable hypotheses, but to assumptions that are at best loosely validated in post-launch analyses after rollout. In the following, generated design knowledge about which design directions the artifact should pursue is presented. Together they should inform the design which aims to answer the research question: *'How should a Decision Support System be designed to improve the generation of testable and valuable product improvement hypotheses in a SaaS startup context'*?

First and most importantly, an effective design should formalize hypotheses in a standardized and intuitive fashion. Otherwise, Product Managers and other stakeholders in the product development process are confusing hypotheses with assumptions again. Hypotheses need to have a clear mapping to concrete target metrics (not just dimensions of metrics such as feature adoption) and experimentation. Otherwise, hypotheses will remain vague. This also means that parameters such as MDE, statistical power, and statistical significance must be included in making a hypothesis distinct from an assumption.

Second, a good design likely nudges Product Managers towards higher quality experimentation designs, already during the course of hypotheses generation. If hypotheses are thought without the means to validate or reject them, it's easier to disconnect experi-

mentation from the hypothesis and choose again a lightweight post-launch analysis (that may not even be statistically significant, with all other flaws such as confirmation bias involved). Showing trade-offs of lower quality experimentation (such as pre-post tests or cohort experiments) as opposed to the gold standard randomized controlled trials, it may educate Product Managers to favor more likely a better experimentation - even though it might be on the cost of waiting longer until receiving significant results.

Third, one could learn from the interviews that Product Managers perceive a strong certainty-speed trade-off under which they operate. Without transparency on the likely duration of an experiment for a given hypothesis, it is safe to assume that higher-quality experimentation is ignored more often than necessary. Often product usage is skewed, where in some product areas more usage and better applicability of higher quality experimentation is possible (e.g. for high-traffic product areas), while in other cases high scientific rigor may not be applicable. Keeping the Product Manager in the loop to decide what is most appropriate given an individual product change's certainty-speed trade-off, will nudge towards higher maturity in experimentation but also gives enough flexibility for startups. Sometimes, lower-quality experimentation given all interpretation problems is still assumed to be a better experimentation choice than a two-year-long running A/B test or no experimentation at all.

Fourth, it is reasonable to assume that Product Managers have knowledge gaps in hypotheses generation and experimentation. Thus, an effective design must help them to learn about these topics and ideally guide them towards changing their way of working. This educational aspect can be accompanied by a logical workflow that is not overwhelming but lets the Product Manager solve sub-problems of hypotheses generation step-by-step.

Fifth, an effective design should not underestimate Product Managers' intuition, creativity, and experience - but leverages these talents where needed. From looking at CEx literature, intuition is inherently seen as unscientific and a flaw in rigorous experimentation. Learning from the interviews, one could see that creativity, intuition, and experience are integral parts of restricting search spaces, relating data to previously found insights, and evaluating an idea's fit on the strategic roadmap. Therefore, an effective design focuses on suggesting opportunities in the software product to improve or potential product changes to enrich the needed capabilities of the human.

Sixth, different artifacts identified such as jobs to be done, ground causes (induced insights from qualitative research), and existing validation techniques (such as user testings and internal sanity checks) should find their logical spot in the course of hypotheses generation. An effective design acknowledges a variety of artifacts' relevance for a hypothesis. This results in a broader definition of hypotheses generation that goes beyond brainstorming a quick idea with an expected effect. Rather, hypotheses generation is considered to be a thoughtful research-informed task. Integrating the existing working mode and processes also results in embedding hypotheses generation and experimentation better into the Product Manager's mental model. As Product Managers are concerned with finding ground causes for user frictions, a DSS builds on their talent to induce concise research insights, but does not automate this complex activity.

Seventh, Product Managers seem to have an understanding of what kind of metrics

meta-categories might be interesting to analyze. Categories such as feature adoption or engagement have been mentioned by multiple Product Managers. As there seem to be patterns in which metrics are looked at primarily, finding a good metrics framework to guide Product Managers further may rationalize the target metric selection for hypotheses generation. Product Managers' mental model involves first exploring the problem space, then the solution space. Thus, an effective design would start to detect problems with the current state of the product first. Then it decides how to optimize the product (solution space) to solve the problem better. This is in fact the reverse logic of having a product change first and then finding a good target metric for it. However, it is about finding a valuable product change to improve a target metric that is currently low-performing.

Eighth, an effective design will need to streamline the process and integrate into the extensive tooling landscape Product Managers have. Making a standalone tool that is in no logical connection to existing tools or processes would be detrimental. Integrations with third-party tools or at least allowing to link documents of other tools should be in focus for an effective design. Arranging information from other tools can also bring order into the product management discipline at the case study company. Interviews have shown that Product Managers work rather individualistically, hence a process guidance that links to other tools and related artifacts can be considered a valuable design choice.

Ninth, Product Managers report that product improvements are conceptualized in a variety of ways. A product can be changed either by launching an entirely new feature or improving existing functionality. An effective design should hence have a flexible concept in place of what a feature (or cluster of features) is. Therefore, an effective design leaves space for the human Product Manager to slice the product into parts that she wants to drive with the product change and select metrics that especially analyze this part that should be improved. Either with a small or a large product change.

Tenth, an effective design should also function as a documentation system and knowledge base. Interview results suggest that product management is also about bringing order to chaos and structuring information for later use. Having hypotheses, corresponding experimentation and their final validation outcome in one place empowers a knowledge base that grows over time and creates a virtuous build-measure-learn cycle for CEx.

## 4.4. Artifact Design Objectives

Having derived many important considerations for the design of the artifact, the initial research question *'How do product management professionals make decisions about which product changes to develop in order to improve the software product?'* can be concretized to guide iterative development.

The interview study has yielded a foundation to come up with an effective design that supports the human Product Manager in the course of generating product improvement hypotheses. It has been found that Product Managers play an integral role in the complex practice of decision-making which product changes to develop. Further, it has been identified that Product Managers might not be experienced in translating these product changes into testable hypotheses and might need guidance to test the hypotheses with

higher-quality experimentation. The role of the Product Manager's creativity, intuition, and experience blended with her domain knowledge is seen not as a flaw in the decision-making process, but as a valuable source to integrate. Following this, a DSS guiding the human Product Manager to more likely translate product changes into testable hypotheses as well as to enrich her horizon to spot product changes that might be even more impactful is a worthwhile design to further pursue.

To relate this design endeavor to literature, a tool that is improving the likelihood of product changes being translated into statistically testable hypotheses achieves the first condition that 'developers treat every planned feature as an experiment' [8] which is seen as the motto of CEx. Further, by (a) spotting areas of the product that should likely be improved, (b) guiding the Product Manager with alternative solutions, and (c) helping to integrate research insights and strategic fit, hypotheses with higher value might be generated. Looking at high rejection rates of product improvement hypotheses in experimentation (cf. [16]), this can ideally yield a higher likelihood of accepted product hypotheses. Ceteris paribus in the context of experimentation, a higher likelihood of product hypotheses being accepted and not rejected creates more business value for the company. Additionally, it also yields more tangible success stories for experimentation. Ultimately, improving the practice of hypotheses generation with a DSS may have the potential to also improve adoption of CEx which is found to be rather moderate (cf. [20]).

The design of the artifact is guided by two main considerations to ensure contributions to both the research landscape and the application domain. First, the knowledge base will be reviewed to build the thesis' DSS on solid ground and benefit from pre-existing designs. Second, the development of the DSS should be guided by Goodness Criteria (GC) that the solution should fulfill to solve the research challenge. Defining these GC early on helps both to keep focus during research and development and to effectively evaluate the artifact in the end. While a review of relevant literature is conducted in the next chapter, the GC are now already defined in this section. For the final assessment, the GC are translated into evaluation criteria that assess each GC:

- GC1: Can the artifact improve the likelihood that features are translated into hypotheses?
- GC2: Can the artifact improve the perceived value of generated hypotheses?
- GC3: Would Product Managers recommend the artifact to other Product Managers (*Applicability*)?
- GC4: Does the artifact suggest accurate improvement opportunities and predict feasibility correctly (*Accuracy*)?
- GC5: Can the artifact empower greater adoption of CEx (*Business Alignment*)?
- GC6-11: Does the artifact perform well in artifact performance criteria according to Product Managers' perception (*Completeness*, *Fit into IT Landscape*, *Learnability*, *Simplicity*, *Traceability*, *Usefulness*)?

# 5. Foundational Theories & Methods

In this chapter, the research reports on core fields in the solution domain in which design knowledge is leveraged for developing an effective artifact. Addressing the phenomenon of hypotheses generation includes various domains from product metrics over hypotheses definition to experimentation planning. Therefore, knowledge from different fields must be incorporated. Moreover, as the tool should also support and educate the Product Manager, Decision Support System (DSS) are briefly introduced.

The remainder of this chapter is structured as follows. First, DSS and their specialties are presented. Second, existing experimentation platforms are presented as well as their core capabilities. Third, conceptual frameworks, hypotheses templates, and prioritization heuristics are discussed with a focus on their core commonalities. Fourth, relevant insights on product analytics, and fifth, overarching metrics frameworks are described. The design knowledge later utilized in the design cycle is already highlighted in each section.

## 5.1. Decision Support Systems

### 5.1.1. Overview

Decision Support System (DSS) are interactive technological systems that assist humans, often people working in decision-making functions, in their tasks to achieve successful decision-making outcomes [78]. Bonczek et al. describe DSS' most effective applicability in those cases where semi-structured problems are present. In these contexts, there is sufficient data and formalizations of rules and heuristics that a computer can play a major role. However, the context is complex enough that human involvement and judgment are essential to the overall quality of decision-making [78].

DSS support human decision makers in an end-to-end decision-making process. Decision-making processes as presented by Herbert A. Simon can be described in multiple decision-making phases. *Intelligence* includes tasks to gather information and to spot worthwhile problems. *Design* involves activities to find good solutions to the given decision-making problem, while *choice* deals with choosing one of many solution options [79]. Simon's three-step logic has been later extended to include *implementation* (realize the decision) and *monitoring* of taken decisions to reason about their success and create learnings [80].

In order to support the human in the multi-step decision-making process, DSS do not act as isolated standalone systems. Rather, many DSS have been found integrating with data sources such as data warehouses or third-party apps to collect information relevant to the decision-making problem. To name an example relevant to the thesis project, Liu et al. [4] present a category of DSS that leverage data mining approaches (cf. Figure

5.1) to make more informed decisions. Leveraging data-mining guided by human input generates the knowledge needed which by itself empowers better decision support.

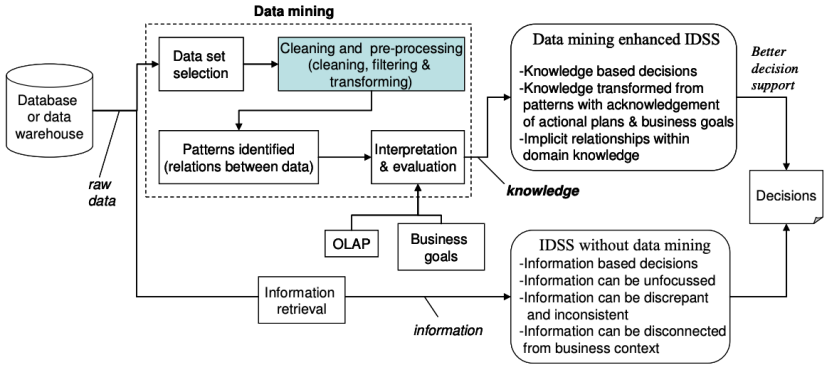



Figure 5.1.: Data Mining Enhanced IDSS for Better Decision Support [4]

### 5.1.2. Leveraged Design Knowledge

From observing literature in the field of DSS, one can learn that DSS support during identifying worthwhile problems, generating alternatives to solve them, choosing one out of many alternatives, implementing the chosen decision, and ideally monitoring a decision's outcome to close the loop. This yields multiple implications for the design of a DSS to empower hypotheses generation.

First, once defining hypotheses generation as a decision-making problem, an effective hypotheses generation DSS should also support the multi-step process of decision-making. This means that a DSS should start with problem identification (e.g. what part of the product to hypothesize an improvement for) until the very end which is the monitoring of the decision made (i.e. the success of a rolled-out product change as defined in a hypothesis). To name a few examples, suggesting alternative product changes might support the Product Manager during the decision-making's design phase. Likewise in the choice phase, an effective DSS supports feasibility considerations, e.g. with simulation functionality of varying hypotheses and experimentation configurations. Moreover, taking implementation and monitoring seriously, the DSS should also integrate with systems that show the downstream implications of the decision made, e.g. trace the decision until the decision's final outcome. In the case of hypotheses generation, this would mean to trace a hypothesis made until its final hypothesis validation in another system. Therefore, an integration of the DSS into an experimentation system is a worthwhile design path.

Second, DSS are often found to apply algorithms for data analysis and integrate third-party information to assist the human decision maker. In the context of a hypotheses generation DSS, having access to data about real user behavior data and external information (e.g. good practices, UI patterns) can potentially help the Product Manager in generating better hypotheses. Offering the human the technical capabilities of data

analysis, patterns can be detected that generate knowledge that the Product Manager can leverage for her decision-making. This knowledge coming from data mining approaches may yield insights on a) where to improve the product, b) which target metric to choose, or c) how to best select statistical properties for experimentation.

## 5.2. Experimentation Platforms

### 5.2.1. Overview

Scientists and practitioners have shed light on experimentation platforms. Those platforms are mostly developed in-house for industry leaders, however, there are also off-the-shelf solutions available. Furthermore, industry leaders such as Spotify also started to offer their solutions to the outside world [81]. Below an overview of experimentation platforms and their specialities is presented. Large parts of this section have been already described in the Master's Seminar paper as well as more thoroughly in a research accompanying blog post by the author[1].

In literature or practitioner blogs, many industry leaders have shared information on their custom experimentation platforms. Companies like Twitter [83], Pinterest [84], Spotify [14, 5], Grab [85] and others are sharing rich information on how they have designed their experimentation systems. In all cases, experimentation platforms are accessible via browser and offer a modern user interface. Necessary data transformations in the background happen with the help of data orchestration tools like Airflow [12] while data is stored in data warehouses or data lakes [12, 14, 83]. The core capabilities of these experimentation platforms are described in the following:

- create experiments (e.g. start date, target metrics)
- define groups of experiment subjects and handle group assignments
- match feature flags (conditional code execution or traffic routing) with experiment groups
- select metrics from a catalog
- estimate sample sizes and other helping hands (e.g. sample ratio mismatch detectors)
- analyze experiment statistics
- orchestrate many concurrent experiments
- analyze experiment data exploratively (beyond significance, e.g. drilling down in countries, segments, device types, ...)
- navigate a user-friendly interface with occasional FAQs

[1] cf. section A.3, especially the blog article about CEx platforms [82]

Platforms let users define experiments with a strong focus on A/B testing in a standardized way - which varies from more technical descriptions in the form of a JSON configuration [84], a domain-specific language [86] or simply via filling out an intuitive experimentation template [5]. Metrics as part of experiments are often organized in centrally curated metrics catalogs which makes finding target metrics for an experiment easy [14], given that some central entity is dealing with metrics curation. Some companies have hundreds or thousands of metrics in place and even support with a metrics recommendation engine [39]. Additionally, helpers such as sample size estimators [5] or special review workflows can be in place to give novice experimenters sparring [87].

### 5.2.2. Leveraged Design Knowledge

As the artifact of this DSR project aims at empowering the generation of statistically testable hypotheses, this also means that the connection to experimentation should be thought with from the beginning. Hypotheses that are never validated or validated insufficiently do neither yield the means to generate knowledge nor do they allow to reason about a product change's success.

First, from looking at experimentation platforms it becomes apparent that an effective design must have a close connection to metrics one can optimize or should control for (cf. Figure 5.2). Companies such as Spotify solve the measurability of product hypotheses by selecting pre-existing metrics from a metrics catalog. In less mature startup contexts, a curated metrics catalog might very likely not exist yet. Therefore, an approach to defining such target metrics in an intuitive yet guided manner should be part of a DSS. Nevertheless, how the metrics are defined (if centrally curated or decentrally created), it seems to be proven that an effective design needs to think in ending up at some form of a metrics catalog or overview.

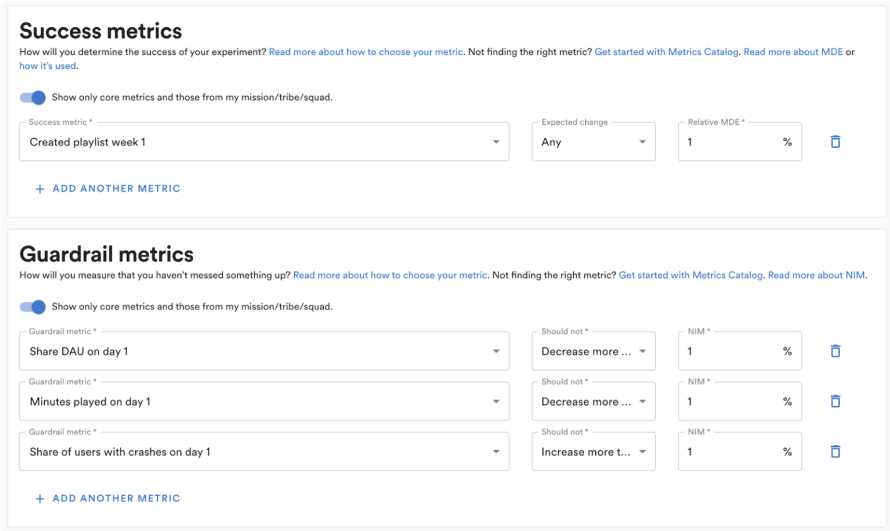



Figure 5.2.: Metrics Catalog at Spotify [5]

Second, feasibility consideration of experimentation designs has its logical place in experimentation platforms. Likewise, sample size estimations (cf. Figure 5.3) for a given hypothesis and experimentation combination may be already useful in the phase of hypotheses generation. If e.g. a button color change with a minor expected increase of 0.1% in a target metric works out for traffic with millions of daily users, such an experiment would take years for a startup to conduct. This would make a hypothesis untestable and thus fail its purpose. However, knowing about the relation of a hypothesis' target metric's expected effects on the estimated time to results, hypotheses might be rethought. A hypothesis could be rethought in a way to include multiple changes at once (e.g. redesigning a whole form instead of a single button color), resulting in a likely higher effect size and hence lower time to results. Further, adjusting desired statistical power and statistical significance gives the Product Manager more control to reduce or increase the robustness and hence time to results for a given hypothesis. As sample size and duration estimations need certain calculation inputs (i.e. a target metric's baseline, current sample sizes, variance, MDE, and statistical properties), an effective design for hypotheses generation must also be designed to have access to this crucial information. A powerful artifact hence tries not only to compile a form of a metrics catalog but needs to be able to collect meta information on a metric.

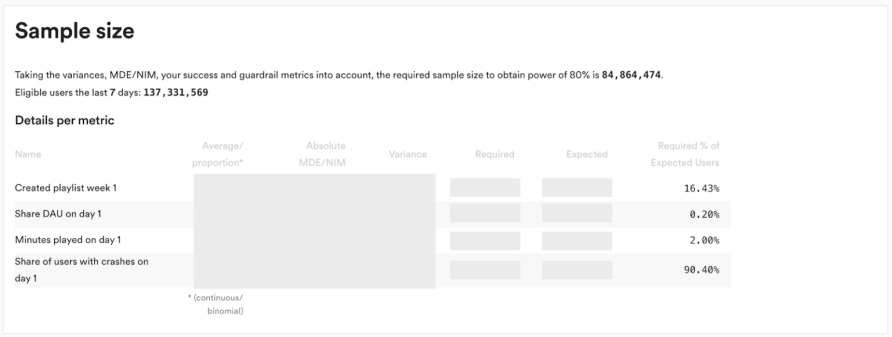



Figure 5.3.: Sample Size Estimations at Spotify [5]

Third, what can also be learned from modern experimentation platforms is that they favor intuitive user interfaces over technical headless tools. Providing templates with input fields, dropdowns, "learn more" links as well as experiment overviews in a timeline perspective have been found in experimentation platforms. Analogously, filling out hypotheses templates with the help of input fields, educational tooltips and a hypotheses roadmap showing hypotheses in their different lifecycles might provide similar value.

## 5.3. Hypotheses Centered Decision-Making

### 5.3.1. Overview

#### Experimentation Process Frameworks

Practitioners and scholars have modeled experimentation processes and the role of hypotheses generation. In general, these process frameworks tend to be heavily influenced by Lean Startup's build-measure-learn cycle [19]. This means that they all include aspects of coming up with solutions, rolling them out while measuring their effects, and in the end, generating learnings about the effectiveness of the solution which yields directions on where to focus next. Thinking in build-measure-learn cycles closes the 'open loop' [88] in software engineering, i.e. it connects the generated learnings of one feature rollout with the hypothesized product changes for following rollouts. While this section does not go into details of respective process frameworks[2] (cf. ESSSDM [89], HYPEX [27], QCD [90], RIGHT [29, 21], or PSR [91]), the author provides a brief overview on the shared perspective on the product improvement hypothesis.

The overall theme in process frameworks is that hypotheses have their initial seed in gap analysis of important metrics, business considerations, or research findings. Gaps between target and actual are seen as a clue for a problem in the current state of the product. This problem in turn provides the opportunity to develop new product changes. Product changes should be formulated as testable hypotheses while their validation creates knowledge about which product changes actually have been effective. This knowledge builds the main decision criterion if features should be rolled out or rolled back. While quantitative experiments after rollout (e.g. A/B tests) have been described in these process frameworks, they are not the only ways to validate hypotheses (or at least assumptions). Also, qualitative-driven approaches (such as user testing interviews) have been mentioned.

#### Hypotheses Templates and Prioritization

When it comes to defining product hypotheses, agencies and tool providers came up with templates regarding what a good hypothesis looks like and which additional aspects to consider. When it comes to defining the important cornerstones for an effective hypothesis, Optimizely proposed a template sentence in the form of *if cause, then effect, because rationale* [91]. For feature experimentation, this means that the *cause* is the treatment (the product change), the *effect* refers to the improvement in one or more target metric(s), and the *rationale* describes why the cause-effect relationship is assumed to be true. The argumentation for the rationale can be made by leveraging insights from qualitative research or data analysis.

Further, practitioners provide frameworks to prioritize different hypotheses in order to distinguish between less and more valuable hypotheses. As testing hypotheses means both developing the necessary product changes and designing experiments, only the most promising hypotheses should be tested - strengthening the necessity for prioritization and

[2]The Master's Seminar paper sheds more light on multiple mentioned process frameworks

ranking. A variety of few-letter acronym frameworks such as ICE (impact, confidence, ease) [92] or PIE (potential, impact, ease) [93] have been proposed. The common notion of all is that hypotheses with higher improvement potential should be prioritized over others as they offer better leverage to move the overall metrics - and hence improve the product. However, also aspects such as the relative importance of different low-performing metrics and the ease of implementing feature changes and experiment design should be considered.

### 5.3.2. Leveraged Design Knowledge

The DSS will build on a variety of input knowledge from the investigated literature. First, a process workflow reflecting the overall theme of the frameworks should be part of an effective design. Many frameworks have been created to better understand the creation and validation of hypotheses and their relation to gap analysis and hypotheses validation. As the pre-study showed, the case study company can also benefit from greater process compliance when it comes to experimentation. Second, thinking in repeating build-measure-learn loops also necessitates a long-term perspective. Therefore, an effective design should ideally incorporate a knowledge extension function that is fed with every new hypothesis validation's learnings. Third, informal validation techniques like qualitative user testing should be part of an effective DSS to empower better hypotheses. If during hypotheses generation, the hypothesis can be already made more robust before development, this will likely lead to a higher likelihood of winning hypotheses. This can likely be achieved by quasi-validating the product change with few users early on e.g. as part of user testing. Fourth, it becomes clear that strategic considerations and research findings should be placed in relation to the hypothesis - so an effective design should also incorporate these dimensions. Not all potential solutions to a problem are of interest to the company, but each hypothesis should be ensured to not conflict with strategic considerations. Moreover, a clear link to research insights can ensure further robustness as the Product Manager needs to base her assumed cause-effect relationship between product change and metrics improvement on research findings.

When it comes to factoring in the important parts of a hypothesis, an effective design should ideally not only support any generation of an *if product change, then effect* hypothesis. But, it should nudge the generation of those hypotheses that are backed by a rationale about its cause-effect relationship. This design motto is followed by downstream artifact design with checkmark clues once all important aspects of the hypothesis are fulfilled.

For the sake of prioritizing hypotheses, an effective design should take ranking potential product improvement opportunities into consideration. A system can ideally spot and suggest opportunities with problems that could be solved by a hypothesized product change. On the other hand, it should have enough flexibility so that the Product Manager can decide for which opportunities to generate hypotheses for while leveraging her context knowledge. Often, the Product Manager can ground a suggested opportunity by leveraging all her domain knowledge and past experience. With the help of estimating the ease of implementation via at least the necessary sample sizes, the Product Manager has an

additional data point to base her prioritization heuristics on.

What all practitioner templates have in common is that they are rather of an exemplary nature and not embedded in a workflow. Naturally, that is what a DSS would digitize, i.e. to support the real-world application of the thinking which is currently mainly concretized in these hypotheses templates.

## 5.4. Product Analytics

### 5.4.1. Overview

The software provider Atlassian describes product analytics as 'the process of analyzing how users engage with a product or service. It enables product teams to track, visualize, and analyze user engagement and behavior data. Teams use this data to improve and optimize a product or service.' [94]. In support of that, the analytics tool Mixpanel defines the term as 'product analytics shows companies what their users really do, not just what they say they do. These are known as revealed behaviors [...].' [95]. The target market for product analytics is SaaS software companies that serve their customers installation-less via the cloud while providing the ease of use of a website [96]. In the majority of cases, SaaS products offer more complex use cases to users than websites. They are thus also more complex to analyze than mainly landing page and call-to-action optimized websites. While traditional web analytics is often constrained due to the identification of the same users in multiple sessions or filtering out robots [97], this is not a critical concern for SaaS products and product analytics. SaaS products typically have sign-up and user authentication methods in place which can be leveraged by product analytics for tracking individual user journeys over time and not just anonymized visitor interactions[3].

The core data set of product analytics tools are event logs which are analyzed with traditional data transformation algorithms to calculate measures that uncover user behavior. The reason is that event logs capture trace data of users' interaction with the software. Due to the nature of event logs, generic concepts can be leveraged for the calculation of quantitative concepts. Jansen's renowned lecture [99] puts web analytics in the tradition of behavioral research that leverages trace data in order to understand human behavior over time. Transaction log analysis of recorded usage events allows the researcher (e.g. the Product Manager) to work with unobtrusive data which is considered quite bias-free as the researcher does not interfere with the subject under study (the user) [99]. Each event itself answers the question of *who did what and when* and allows for diverse ways of metrics calculation, segmentation, path, and conversion analysis [97]. Besides information on time and user, higher resolution (and hence depth for analysis) to events is given via additional properties (key-value pairs) that give additional context e.g. about the user, the device, location, and more [100].

Beyond just analyzing data as it is, there exist commercial product analytics tools on the market that have functionality that moves beyond mere data analysis toward bringing context to data. The reason is that they seem to see value in moving closer to

[3] cf. Pendo API where users are identified with a unique ID [98]

finding patterns in data which decreases the gap to actionable insights for decision-making. Leading companies seem to all offer similar base functionality ranging from funnels, click-through paths, and traditional analytics to analyze product usage (cf. Mixpanel [101], Amplitude [102], or PostHog [50]). Also, they provide ways for Product Managers to drill in the data or only view data of parts of the software. Beyond these standard features, they seem to move beyond as-is analysis towards providing contextual insights to the data, e.g. via identifying anomalies [103, 104] using approaches such as machine learning. Yet, while machine learning has shown already great potential in the domain, Gupta et al. conclude that in the early 2020s, human-independent algorithms are 'still very imperfect' [105].

### 5.4.2. Leveraged Design Knowledge

Product analytics is seen as a data analysis umbrella to better understand product (area) performance by investigating behavioral traces. For the sake of hypotheses generation, it results in the implication that a good design ideally leverages event data of users as it is found to be key in evaluating product performance and compiling worthwhile metrics. Building on event logs which seem to be ubiquitous in the industry also allows for a greater applicability of the design. Doing so, it does not solely restrict the artifact's applicability to the case study company as a widely accepted data format is used.

```
{
   "timestamp":"2022-12-01 00:00:00",
   "user_id":"421ea7f41b6c4dfb8cc3f87f71bfa5af",
   "name":"user_profile.picture_uploaded",
   "properties":{
      "prop1":12,
      "prop2":true,
      "prop3":"https://company.com/profile/123"
   }
}
```

Listing 5.1: Event Example

As product analytics solutions usually offer customization options (e.g. selections of a few events) and a cherry-picking of metrics for a given context, the DSR artifact should also be customizable to select certain events for a given product area to improve. Further, it should suggest thinking about different metrics to define hypotheses for - but it should not treat all of them as mandatory. Sometimes deep engagement with a feature is a sign of good product performance (proxying that users see value), in some other cases it may be irrelevant. Moreover, bringing context into analyzing metrics to not only display as-is values but also anomalies (e.g. in a simplistic form such as a gap analysis between expected and real) seems to be a worthwhile indicator to spot improvement potentials. To not introduce unwanted complexity while analyzing usage events, the researcher is excluding the segmentation via event properties.

## 5.5. Product Metrics Frameworks

### 5.5.1. Overview

Product analytics builds on industry standard measurement concepts which are picked and customized for a certain business context. The rationale is that due to the core logic of how web applications work and a common narrative of success indicators, generic measurement concepts could be derived. Already in the 2000s, the Web Analytics Association provided definitions of often-used measurement concepts such as unique visitors, click-through, bounce rates, or engagement measures (e.g. page views/session) [97]. In support of that, Chaffey and Patron (2012) [106] describe overlapping measurement concepts of digital analytics that go in the same direction. Their list includes e.g. analyzing user acquisition (e.g. cost-per-click), reach measures (unique visitors), usage (e.g. engagement or conversion rates), satisfaction (e.g. net promoter score), or customer-value measures (e.g. revenue per visit). Systematic literature reviews prove the notion that a common set of measures appears again and again [107]. However, as the importance and exact design of metrics is dependent on the business context, they are picked or adapted accordingly. For instance, average order value might play a large role in e-commerce while page depth or returning visitor ratio might be important for other pages [99].

To bring order into all these metrics, multiple overarching metrics frameworks have been proposed as core mental models to relate specific metrics to. Due to web applications' common foundations and measurements, a generic abstract layer can be applied on top of metrics to analyze most software. To name two influential frameworks, McClure's Pirate Metric Framework [108] and Google's HEART framework [109] have received major attention in the practitioner field. First, the Pirate Metric[4] framework defines five primary optimization areas in a software product. Ranging from acquisition, activation, retention, referral, and revenue it attempts to cover major aspects of most web applications. The framework likely has a greater applicability for websites and online shops. Individual metrics (e.g. the metric cost-per-click to the dimension acquisition) can be related to each optimization layer.

**McClure's Pirate Metrics AARRR** [108]:

- Acquisition: Users come to the site from various channels
- Activation: Users enjoy 1st visit and experience a happy moment
- Retention: Users come back, visit the site multiple times
- Referral: Users like the product enough to refer to others
- Revenue: Users conduct some monetization behavior

Second, the HEART framework - which is based on around 20 successful projects at Google - has been popularized and widely used since 2010. Similarly to Pirate metrics, it

[4]Named after its acronym AARRR

describes optimization potential for a software along five categories, namely happiness, engagement, adoption, retention, and task success. While the categories are constructed slightly differently, the framework also partly overlaps with the Pirate framework (e.g. retention or the similarity of activation and adoption).

**Rodden et al.'s HEART framework** [109]:

- Happiness: Attitudinal opinion towards solution (ease of use, referral)
- Engagement: Users interaction with the solution (frequency, intensity, depth)
- Adoption: Users starting to use the solution
- Retention: Users coming back to use the solution regularly
- Task Success: Are users achieving their tasks (e.g. following the optimal path)

Besides hypotheses generation as a result of the analysis of the optimization layers' (HEART or AARRR) metrics, both frameworks mention also the role of experimental practices such as A/B testing for validating the predicted success on target metrics. Having meta concepts for improvement levers in place (e.g. adoption, retention, engagement, ...) as well as analyzing corresponding metrics offers a reasonable starting point for identifying product improvement opportunities.

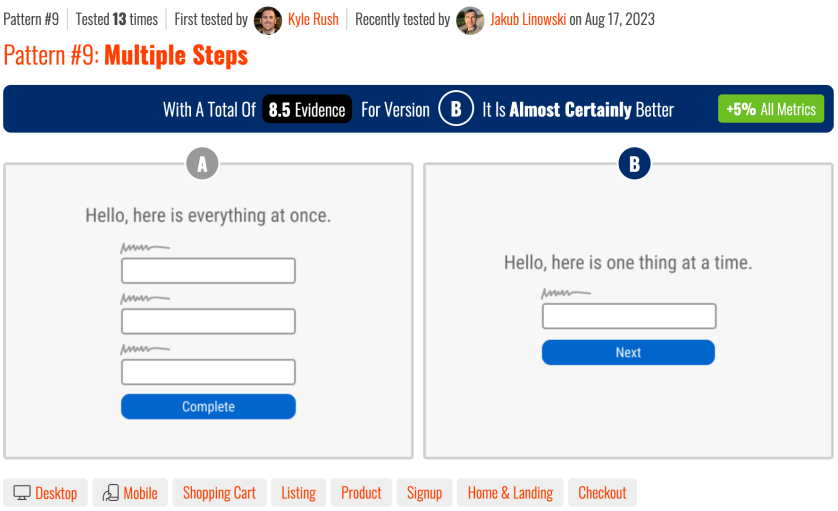



Figure 5.4.: GoodUI.org's Product Improvement Patterns [6]

Besides generic ideas on how to aim at improving certain metrics categories of such a framework, there are also attempts on the practitioner landscape to find patterns of what works well to improve certain categories (validated through multiple A/B test successes), e.g. on GoodUI [110]. To illustrate the kind of winning A/B test patterns that are shared on GoodUI, an example is provided in Figure 5.4.

### 5.5.2. Leveraged Design Knowledge

From looking at metrics frameworks, it is reasonable to include a proven framework logic for an effective DSS to empower hypotheses generation. The HEART framework has been found reasonable for developing the artifact as it seems to be more applicable to a B2B SaaS product that builds on subscriptions and not referrals or in-app revenue. As it has been found paramount to relate some metrics to each category of the framework (e.g. weekly retention rates to the retention category), a small selection of metrics to choose from should be part of a hypotheses generation DSS.

However, not only analyzing worthwhile metrics for rationally optimizing a product area is interesting. As these meta-categories such as engagement or retention also have implications for restricting the search space for potential product changes, a stack of potential product changes targeting each of the framework's metrics dimensions can be integrated. The artifact design thus attempts to compile also a selection of best practice product changes, Large Language Model (LLM) generated ideas, and winning product changes from A/B tests to improve a certain metrics category. To illustrate this, given an improvement opportunity in the dimension of retention is spotted, the Product Manager may get inspired by alternatives to change the product which especially are tailored to improve retention.

# 6. Artifact Design

This chapter presents the final artifact in detail as well as the two intermediate iterations. In order to satisfy the claim of providing insights for management and technology audiences, the high-level design as well as implementation details are covered.

## 6.1. The Artifact - HypGen

### 6.1.1. Core Idea

#### The Hypothesis

The thesis' approach to improving hypotheses generation demands for a more holistic portrayal of *the hypothesis*. If a hypothesis is defined in isolation, it may not be grounded in research, aligned with product strategy, or feasible to validate in experimentation. All of which would either decrease the value of a hypothesis (e.g. not fitting product strategy, not solving a user problem) or question its ultimate validation in an experiment (e.g. waiting a year to validate the hypothesis). On the one hand, the literature review has shown how process frameworks and hypotheses templates contribute to an improved understanding of effective hypotheses generation in the context of CEx. On the other hand, the interview study showed the missing formalization of hypotheses as well as their disconnect from experimentation in the B2B company in scope. To be embedded in the artifact's design, the author has aimed for an extended definition of the hypothesis in the context of CEx. The paper takes over the base hypothesis logic found in the literature, i.e. following a pattern of 'if *product change*, then *improvement in target metric*' - with optionally adding *because of reason* (cf. [91]). To go a few steps further, the thesis builds a greater emphasis on a) the connection of the hypothesis to experimentation already, b) the connection of the proposed product change to user research, strategic fit, potential alternative solutions, and product health, as well as c) thinking about hypothesis validation's certainty-speed trade-off (cf. 6.1).

#### HypGen: At a Glance

The DSR artifact is called HypGen, which is a short form of **Hyp**otheses **Gen**eration. HypGen aims to support SaaS Product Managers in (a) analyzing key SaaS product health metrics to (b) spot impactful improvement opportunities in order to (c) improve the product with statistically testable product hypotheses which (d) can be statistically validated in experiments after release.

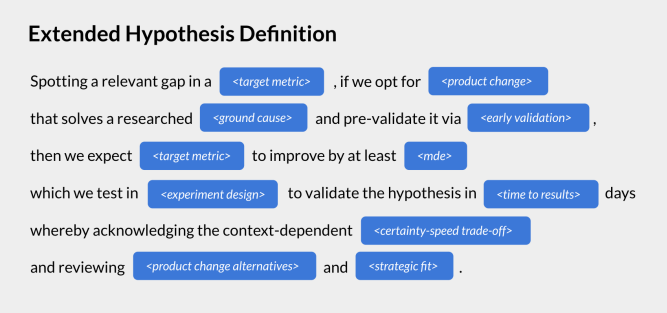



Figure 6.1.: Extended Hypothesis Definition

Leveraging a software company's real usage event and user survey data[1], Product Managers can ultimately come up with product improvement hypotheses that are:

- *impactful*, i.e. hypotheses that target at an opportunity where a large gap to expected target metrics was identified
- feasible, i.e. hypotheses that can be statistically tested in a reasonable time frame with a given experimentation design
- *set-up for success*, i.e. hypotheses that are pre-validated, and grounded in research insights while being inspired by alternative product changes

### 6.1.2. Final Design

The following chapter highlights the core capabilities and functionality of HypGen. To stay within the scope of the paper, the author documents and briefly describes the core design elements and user flow. Links to the code repository can be found in section A.1. Aspects that appear all over again like in-app education (e.g. the ubiquitous appearance of tooltips or context views) are not separately described. Yet they can be seen and are partly described when presenting primary functionality. As the prototype has been used with real-world data, the images to present the design are either visualized with the help of the analogy of a generic project management software or are depicted in an anonymized fashion.

Towards the end of the final design's overview, the two intermediate designs are illustrated. The two intermediate designs have received feedback from subjects of the study and have informed adjustments to HypGen which ended up in the final design.

#### Overall Design Composition

HypGen can be accessed via the browser and is designed to have a modern user interface (cf. Figure 6.2). Its core navigation elements are *Product Area*, i.e. one can see how Product

[1]To not unnecessarily increase the complexity of the thesis, real world event and survey data has been provided as .csv files and is not ingested via a live connection to their product analytics software

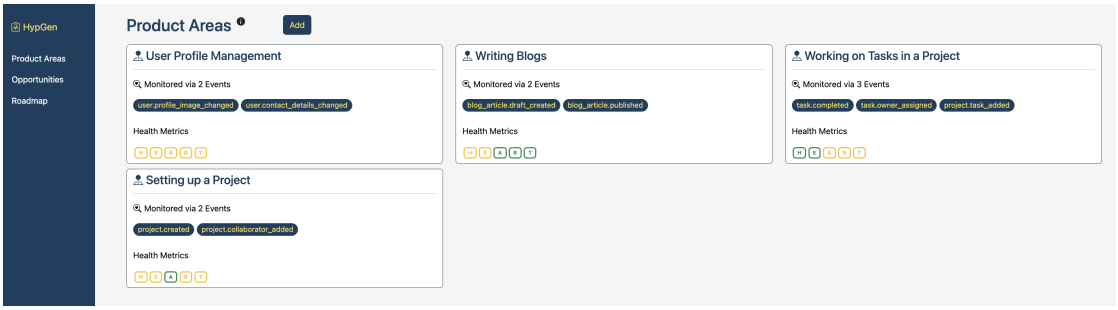



Figure 6.2.: Design: Overall User Interface

Managers have decomposed the SaaS product into its logical components. Clicking on a single *Product Area* opens a detail view of the sub-part of the software product. Second, *Opportunities* lists all the ranked improvement opportunities that HypGen has identified, with a drill-down to sub-select the relevant ones for a single product area. Third, *Roadmap* is an overview of hypotheses according to their different lifecycle stages which is described later on. Clicking on a single hypothesis on the *Roadmap* view allows the user to access the detail page per hypothesis.

### Product Area & Health Analytics

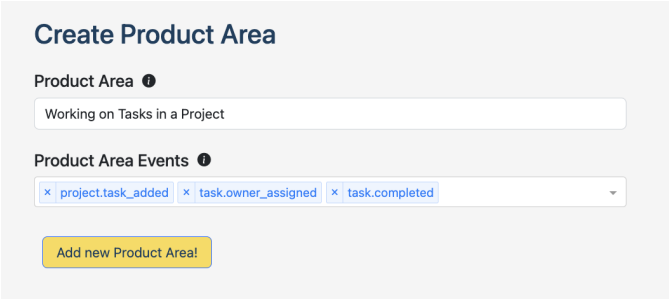



Figure 6.3.: Design: Create Product Area

Software products as a whole are likewise hard to analyze and optimize. Breaking down the SaaS product into logical parts allows the Product Manager to target analyses at specific existing features or groups of features (cf. Figure 6.3). The Product Manager selects a set of available usage tracking events that belong to this product area, hence creating a digital twin. Using the analogy of a project management tool, a product area could be functionality that is concerned with working on a project (setting up tasks, assigning task owners, and completing tasks). Product Managers would select appropriate events to track usage in the frame of engaging with the 'Working on Tasks in a Project' feature (group). The events to select from are insoured from a real event tracking dataset. It depends on the Product Manager's domain knowledge which events are important for

selection and on which level of granularity the product should be decomposed.

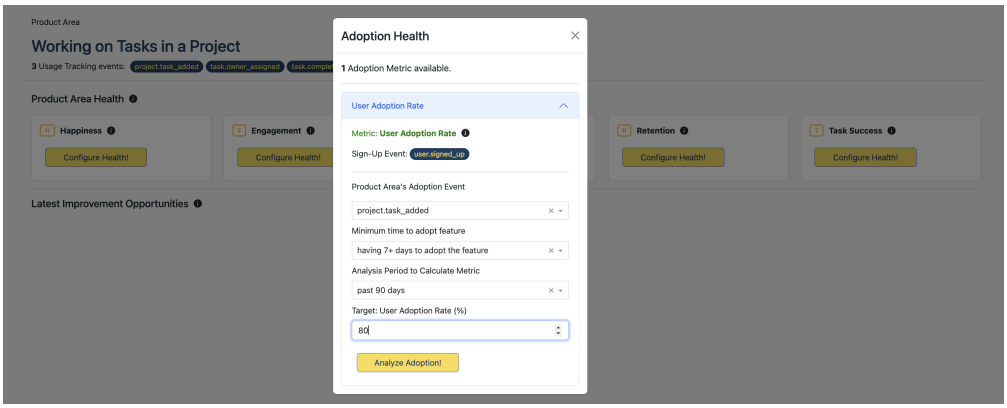



Figure 6.4.: Design: Configure Health Metrics

Continuing with the product area just set up, HypGen helps the Product Manager to find potential product improvement opportunities along five product health dimensions (one for each letter of the HEART framework, cf. Figure 6.4). Exemplary use cases to illustrate the dimensions can be found below:

- It could be that a product area might have only little user satisfaction (maybe caused due to bugs, bad user experience, ...) which one can find out by analyzing *Happiness*
- *Engagement* with the product area could be lower than expected. Users might just superficially engage with the product area but the Product Manager perceives a higher engagement depth as desirable
- *Adoption* might be an opportunity to improve, i.e. fewer people than expected have started to use core functionality of the product area
- *Retention* might cause issues for the given product area. Users may have adopted the product area, but at some point do not return anymore
- Users might not be fulfilling their tasks successfully (coined *Task Success*) with the product area, e.g. many users fail to complete a certain multi-step action

Having transparency along which dimensions to potentially improve her product area, the Product Manager can decide which dimensions are most important. Having selected a metric belonging to a health dimension to improve, the Product Manager provides reasonable target values. Then, HypGen automatically analyzes event or survey data (in the case of happiness) to generate improvement opportunities based on target-actual comparison. HypGen also allows for reconfiguring health analyses and rerunning the opportunity detection.

### Improvement Opportunities

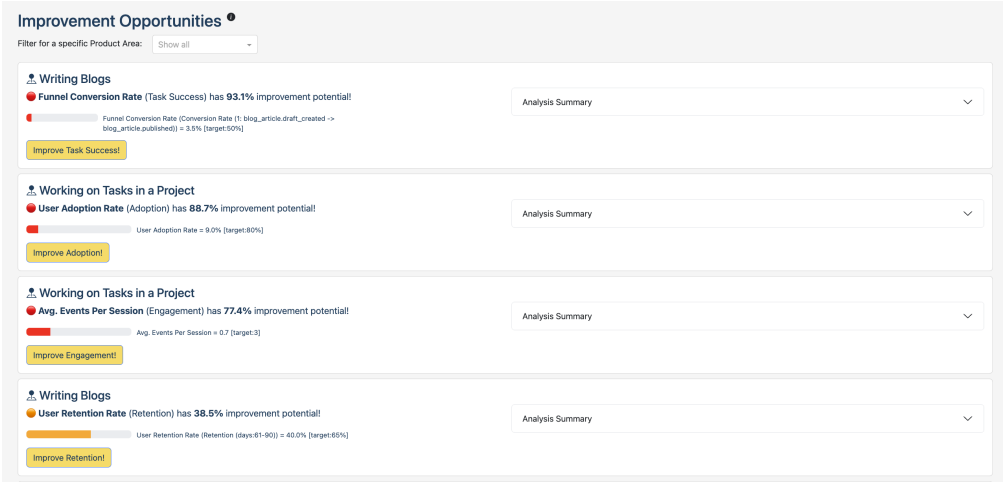



Figure 6.5.: Design: Improvement Opportunities

Based on health metrics configuration and gap analysis, HypGen is suggesting worthwhile improvement opportunities (cf. Figure 6.5). Improvement opportunities are aspects of product health that are underserved by the current state of the product area. Targeting a sub-optimal area is perceived in many cases as yielding more likely large improvements than already quite satisfying health categories. For exceptions to the rule, the Product Manager can factor in her domain knowledge to reason about the suggested ranking. An improvement opportunity of e.g. 70% in user adoption, would mean that the Product Manager assumes that a reasonable target for user adoption for a given product area is way higher than the current adoption rate. Thus, generating a hypothesis to improve adoption (e.g. via a product change that eases the discoverability of the feature) might be valuable as there is plenty of room for improvement. To understand each improvement opportunity better, the Product Manager can unfold an analysis summary and dive in deeper. Diving in deeper also allows to view an analysis graph where gaps are visualized. In the background, the analysis to spot gaps has also collected meta-information such as daily sample sizes as well as metric variances which are later of use for experimentation feasibility predictions.

Browsing the different product improvement opportunities, the Product Manager can simply create a product improvement hypothesis now with just a click on the yellow button (cf. e.g. the 'Improve Adoption!' button in Figure 6.5). Doing so also means having selected the target metric to improve as well as its meta-information.

### Hypothesis (& More)

Having decided which opportunity to act on, the core of the hypotheses generation process starts (cf. Figure 6.6). The Product Manager can check again the sub-optimal target

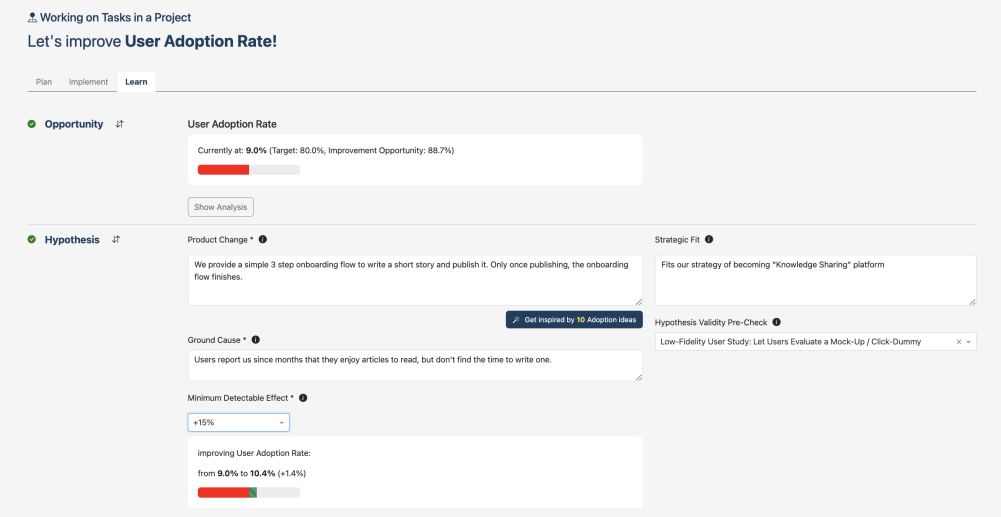



Figure 6.6.: Design: Hypothesis

metric (e.g. a specific product area's adoption rate) and can brainstorm potential product changes that might drive the target metric. In the case of adoption, this would mean that something is changed in the product that is hypothesized to lead to relatively more users adopting the feature than without the product change. Further elements that are strengthening the hypothesis before rollout are ensuring that the product change is valuable. First, the change should be aligned with strategic considerations of the company. Second, the template ensures that a change is not just an idea of the Product Manager, but that it satisfies a clear ground cause of why this change is solving a user friction or problem (e.g. insights coming from qualitative research). Improving the value of a proposed product change, the hypothesis is ideally also pre-validated before implementing the code. HypGen nudges the Product Manager to think about ways to already test the hypothesis before handing it over to the development team. Finally, the Product Manager needs to decide on a Minimum Detectable Effect (MDE), which should describe how far the Product Manager believes the proposed change can drive the target metric at least.

The Product Manager is not left alone with brainstorming ideas which means being constrained by her intuition, creativity, and experience alone. However, the Product Manager can be inspired by potential product changes that are meant to drive the target metric (cf. Figure 6.7). This is meant to ensure that the Product Manager is less prone to biases, such as confirmation bias. Own ideas might be challenged with alternative solution approaches, hence enriching the potentially biased solution space with an outsider perspective. The Product Manager can also check best practices, previously winning hypotheses for similar metrics or suggestions generated by Generative AI[2]. The Product Manager can pick these suggestions as they are, or merge them with their ideas in mind.

[2]AI generated suggestions are not dynamically generated, but a list of AI generated product hypotheses per product health category (HEART) has been provided

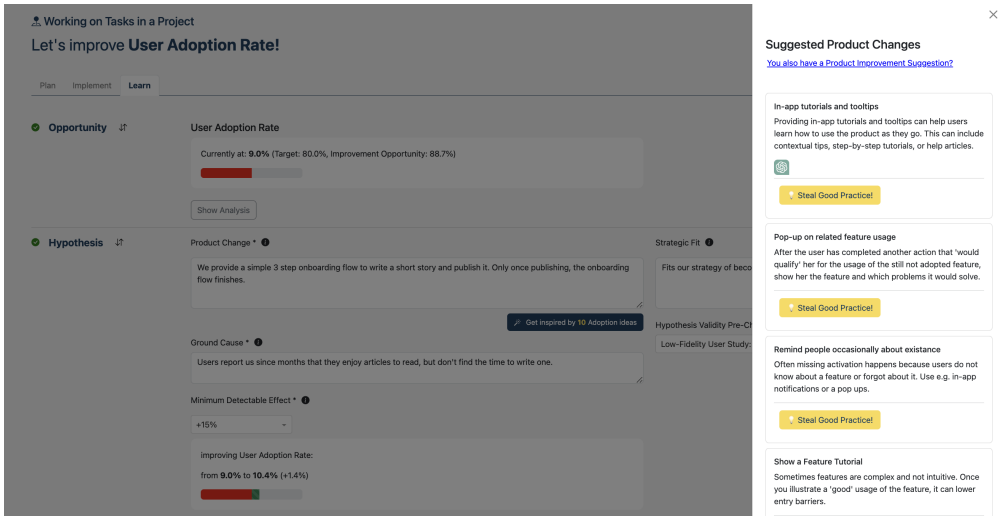



Figure 6.7.: Design: Product Change Suggestions

Based on a carefully generated hypothesis, HypGen also includes planning of the experiment (cf. Figure 6.8). Without the experiment in mind, a hypothesis could be a) never tested or b) not feasible to test[3]. HypGen enables the user to explore multiple experimentation approaches of varying scientific rigor[4], illustrating pros and cons as well as their primary applications in a context menu. Further, it allows to set statistical parameters and in conjunction with already available information (target metric's current baseline, sample sizes, and variance) as well as the desired MDE of the hypothesis, an estimation of experiment duration can be conducted. This makes it transparent for the Product Manager if a test approach of high scientific rigor is desirable, or if rigor must be lowered to accelerate experimentation speed, although at the cost of decreased certainty. The other way around, the system also shows the Product Manager when it is possible to aim for higher experimentation rigor than expected. This is key because in certain product areas with lower usage, often highly reliable experiments can simply not be run. The impossibility of experimentation at some area of the product might be taken as an excuse to not engage in experimentation at all, so HypGen can create transparency where higher rigor experimentation might work - and where it does not. HypGen also gives visual clues on how to lower the expected time to results and is transparent about the trade-offs the Product Manager will opt in once adjusting best practice statistical properties. Once planning is finished, the experiment can be exported to PostHog, an external experimentation system. Fetching current experimentation data, HypGen can also display information about an experiment running[5].

[3] E.g. a randomized controlled experiment with 2 groups might be not feasible for a product area's target metric if it takes years until the hypothesis can be approved or rejected

[4] Four approaches available: Randomized Controlled Experiments, Pre-Post Tests, Segmentation Tests, Multi-Armed Bandit

[5] In a rather rudimentary fashion, this was not in the larger focus of this research

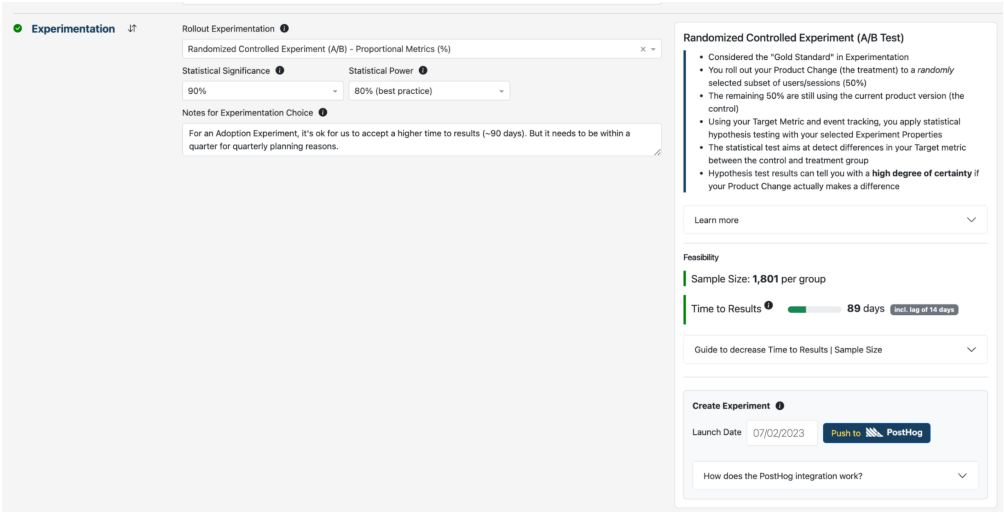



Figure 6.8.: Design: Experimentation

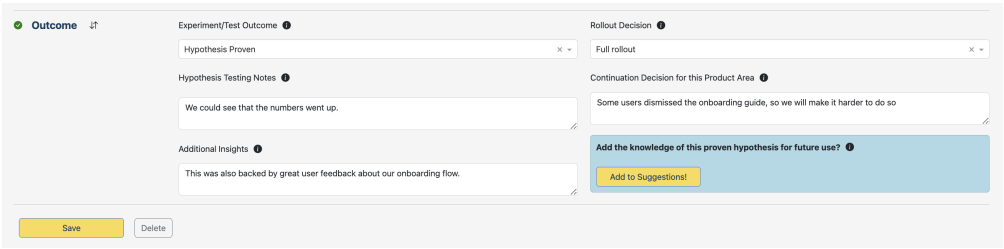



Figure 6.9.: Design: Outcome

Once the experiment is over, the outcome of the experiment can be documented in HypGen (cf. Figure 6.9). This has two purposes. First, it closes the build-measure-learn loop and allows to trace each decision about a hypothesis to its final outcome. Second, documenting a successful outcome can be used to improve suggestions for product changes in the future. Once another Product Manager is working towards improving a similar metric for the next time, the learnings from a previously validated hypothesis will be used to suggest an alternative product change for future hypotheses generation. This is meant to create a knowledge base of potentially effective product changes.

The roadmap view shows hypotheses in its different lifecycle stages and which product area they belong to (cf. Figure 6.10). Some hypotheses are still in the *plan* phase (generating hypothesis, pre-validation, ...), some other hypotheses might be in the *implement* phase (e.g. the functionality is currently developed), or in the *learn* phase (where the outcome is collected). This ensures transparency and also highlights accepted and rejected hypotheses as well as chosen experimentation at one glance. In this view, the hypotheses are written in a textual form, resembling the notion of the extended hypothesis definition template from above - however with a key focus on the product change, its expected effect

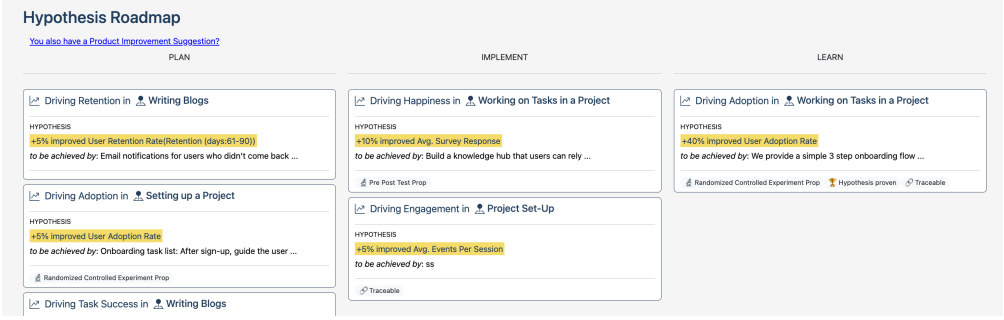



Figure 6.10.: Design: Hypotheses Roadmap

on a target metric, and selected key information.

### 6.1.3. Intermediate Designs

As usual for DSR artifacts, HypGen has been developed in multiple increments. The final version has just been presented. Preceding, two intermediate versions of HypGen have been developed and presented to product management professionals at the case study company. While the final design was evaluated in a rigorous scientific setting (cf. 7), for the two intermediary designs, qualitative feedback for further development has been collected. Below, the reader finds a quick presentation of intermediate designs, the feedback it received, and which design objectives it has yielded for the next version.

#### Version 1: yaml-based Config and Compiled Web Page

In the very first iteration, the researcher focused on creating a minimum viable research prototype (cf. Figure 6.11). Users needed to change a yaml configuration where they could define a feature. The user needed to specify target metrics to analyze along three dimensions of the HEART framework (adoption, task success, and retention). For each feature, the user needed to decide which usage events to include (possible events have been given as text strings in a code comment). For each metric, the user needed to decide desirable targets.

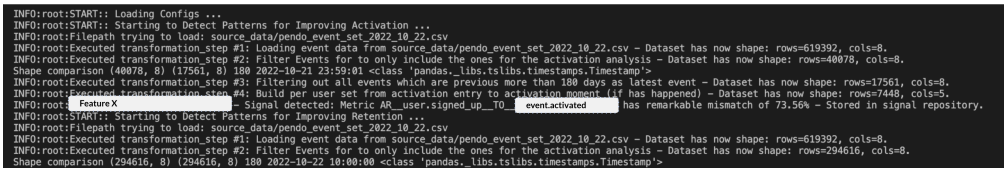



Figure 6.11.: v1: Yaml-Spec and CLI

Once executing a script on the command line, the metrics have been sequentially analyzed. Finally, a web page (cf. Figure 6.12) was compiled that allows the Product Manager to explore improvement potentials. Also, the user could already check the

feasibility of one type of experimentation (A/B test for proportional metrics). Parameters such as statistical power, confidence level, and effect size could not yet be changed but were kept at default levels.

Notably, in this version of the software, providing the product change of the hypothesis as well as its minimum effect on the target metric was not yet considered. The design at this point was largely aimed at detecting improvement opportunities with dedicated target metrics and generically connecting those to experimentation. This allowed to reason about both the opportunity for a hypothesis and the ease of feasibility to validate the hypothesis with A/B testing solely.

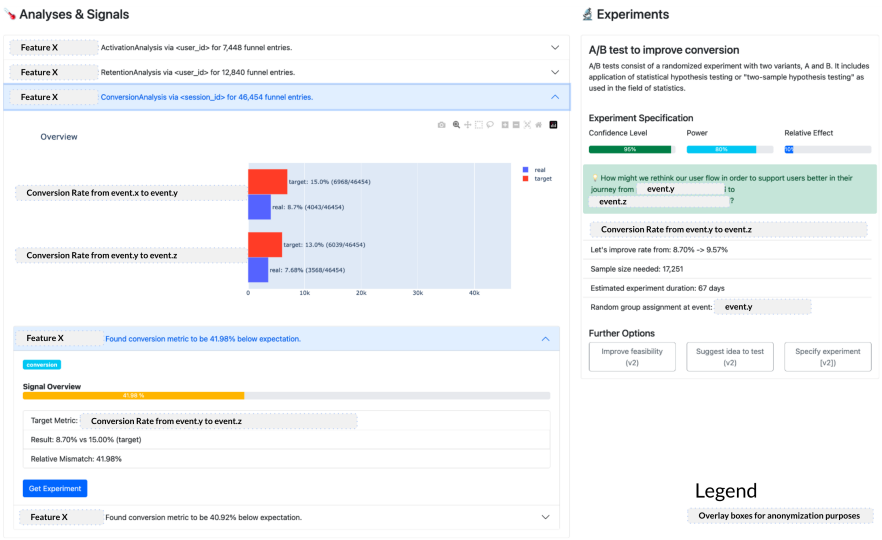



Figure 6.12.: v1: User-Interface (read-only)

Two subjects have been part of the intermediate evaluation. They have enjoyed the connection of target metrics and prediction of experimentation duration as it builds the bridge for Product Managers from opportunity identification to experimentation already. The yaml configuration was understood to be prototypical as it is likely error-prone for production use as non-technical personnel would operate the tool. Both subjects brought up the role of low-fidelity user testing that the UX Researcher or Designer would ideally conduct before the rollout to be part of the solution. For startups, user testing with a few users is a rather cheap way of quasi-validating hypotheses. A subject pointed out that it would be nice to receive information on the selling points of experimentation as well as what an experiment means in terms of setup. Further, it was mentioned that such a tool should better include a long-term view (such as a roadmap) to trace hypotheses as tested with experiments to their outcome, to see if they have been successful or not. Moreover, adding ways to interact with the software to add insights from user research to tell potential reasons why a metric is low performing were mentioned as good ways to proceed.

### Version 2: HypGen as Web App with Hypotheses Generation Capabilities

For the second iteration, major efforts have been taken to level up HypGen to a cooperative web-based solution that Product Managers can better interact with, moving it closer to a DSS. It allowed Product Managers to create custom product areas to break down the product into logical pieces and to have intuitive ways to configure target metrics. There was no need anymore for ex ante knowledge about which event types are currently available (cf. Figure 6.13).

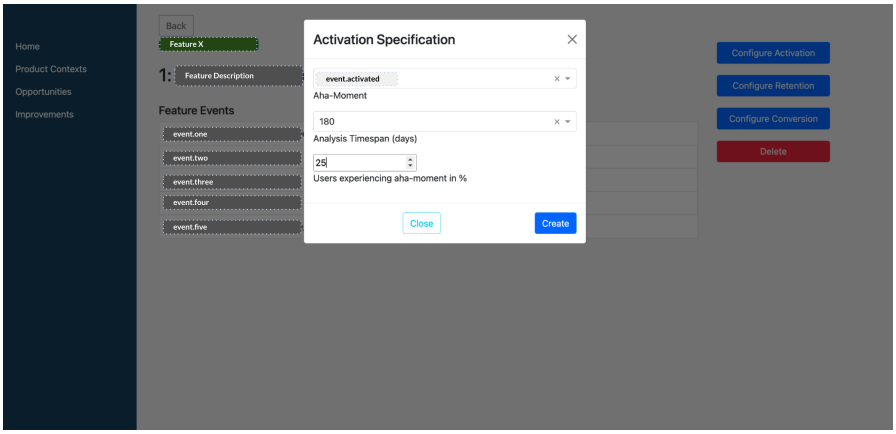



Figure 6.13.: v2: Configurable Health Analyses for Custom Product Area

Being now able to conduct user-specified health analyses, the user could browse a list of ranked improvement opportunities (cf. Figure 6.14). Once opting for an improvement opportunity, the target metric and analysis results are loaded to create a product improvement based on it. The product improvement can be understood as the hypothesized product change, its effect on the just chosen target metric, and the experimentation selected.

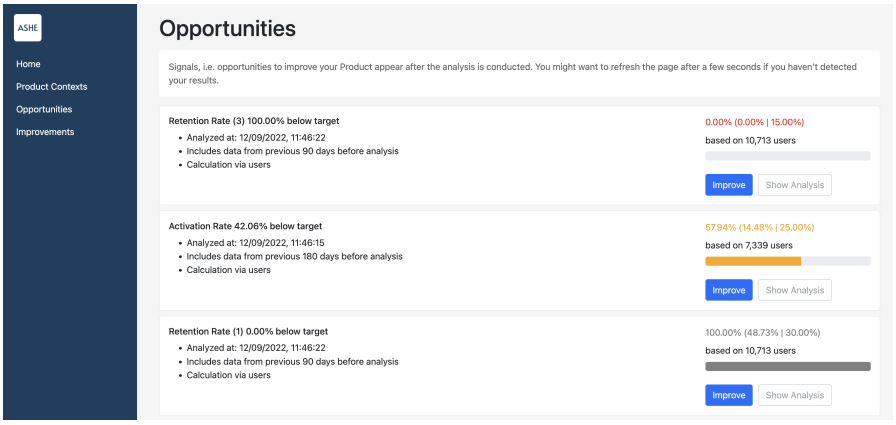



Figure 6.14.: v2: Revamped Opportunity Ranking

Now, it was possible to specify the hypothesis, i.e. how the product will be changed in order to improve the overall target metric (cf. Figure 6.15). Text fields for control and treatment have been meant to be a good comparison to derive the product change (delta between control and treatment). Already now, a brainstorming functionality has been included where Product Managers can be inspired by suggestions for potential product changes to test the applicability of the logic.

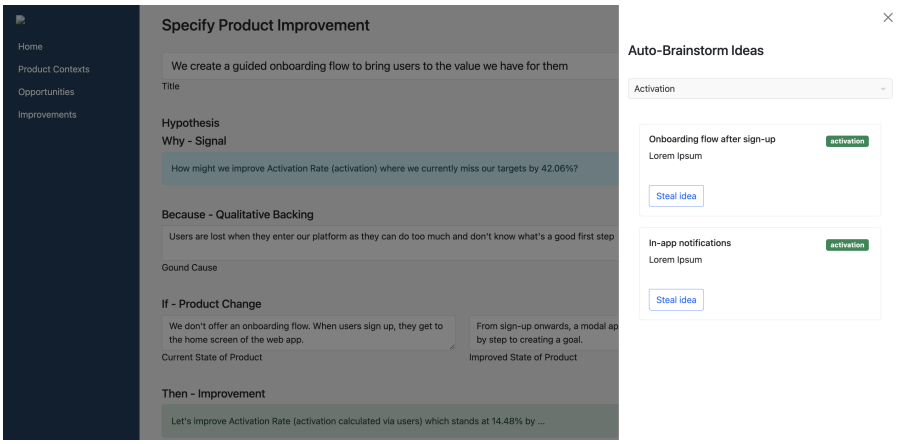



Figure 6.15.: v2: Hypothesis Template and Product Change Suggestions

To summarize, HypGen has now been transferred to an interactive web application. Beyond what can be seen in the above images, HypGen has allowed the Product Manager to create and review product areas, specify health analyses (still restricted to the 3 proportional metric types from the first iteration), analyze metrics, and detect improvement opportunities. Also, it allowed to specify hypotheses and experimentation on a more granular level. When it comes to experimentation, HypGen nudged the user to think both about qualitative validations before rollout and post-rollout experimentation with real usage data. While the first determines to mitigate the risk of a hypothesis early on (by quasi-validating it informally or with low samples), post-rollout experimentation nudges the user to validate a hypothesis with real usage data in high-rigor experimentation. While both are reasonable, the Product Manager could decide on the applicability given a specific context. The system offered different experimentation approaches for each kind (pre and post-rollout).

Again two subjects have used and feedbacked the prototype. They appreciated the workflow from opportunity identification over specifying the hypothesis to choosing appropriate means to validate the hypothesis. HypGen's design was already seen to potentially change their way of working as it fitted their reasoning for how they could work better with respect to data-driven product management. They especially liked the pragmatism that the tool is applying. According to them, allowing to select experimentation with lower rigor is reasonable as lower-quality hypothesis validation may be deemed good enough or the only practicable approach in some situations. Both subjects highlighted the need for a better user experience as it involves quite complex tasks for a not-yet-experienced type

of user.

Missing functionality to onboard the user to work with HypGen and how to operate it have been mentioned. Subjects also raised interesting aspects to reflect on, e.g. (a) if the ranking of improvement opportunities is always accurate regarding proxying what to focus on or (b) what such a tool would mean for partly moving away from their existing documentation tool where they document feature definitions.

The last iteration was then informed by the overall backing that the general design is worthwhile, yet the usability needs to be improved. Thus, the third iteration has been strongly focused on improving user experience, moving beyond just rudimentary functionality in some parts. This meant providing real product change suggestions, extending health analysis to all categories of the HEART framework, and moving beyond proportional metrics. Also, an onboarding guide to the tool has been developed.

## 6.2. System Design

This section will now focus on the technical implementation of HypGen. First, the core technologies and tools used are presented. This is followed by a 4+1 view depiction (cf. [111]) on the software system to examine it from a variety of perspectives.

### 6.2.1. Tech Stack

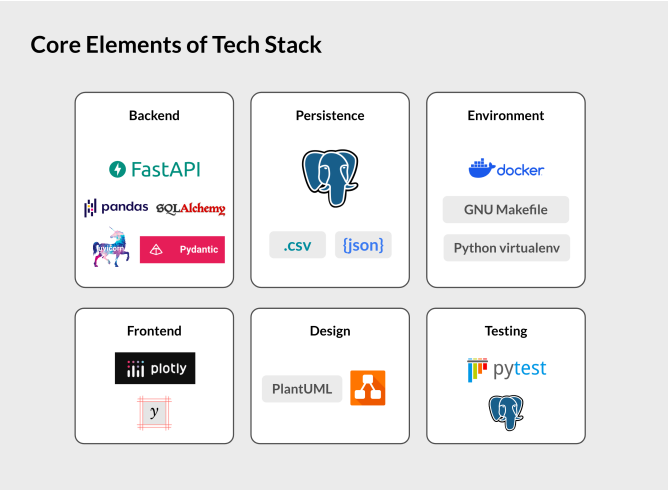



Figure 6.16.: HypGen's Tech Stack

The DSS HypGen is developed leveraging an extensive technological stack[6] (cf. Figure 6.16). The core elements are described here, while in the Appendix (cf. section A.8), the most important Architectural Design Decisions (ADD) are presented which have led to

[6]Logos [112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122] are extracted from the online sources that are also cited in the following tech stack description (cf. image citation advised in [123])

the final tech stack. In general, the tech stack is python-centric, whereby using *python 3.10.0* [124].

**Backend:** The backend is developed with fastapi [112], a python backend framework that makes use of pydantic [116] for data validations (e.g. for API requests). For communicating with a database, sqlalchemy [114] is used as ORM (object relational mapper). Extensively leveraging python's dataclasses [125] allows HypGen to use less boilerplate code for python classes, and most importantly, it allows to create frozen dataclasses to define value objects needed to implement Domain-Driven Design (DDD). Further, pandas [113] is used as HypGen involves data analysis and metrics calculation, whereas pandas' capabilities are used for data transformations. To run the fastapi server, uvicorn [115] is used as ASGI web server implementation for Python applications.

**Frontend:** HypGen's frontend that is accessible in the browser is developed with plotly Dash [117]. Dash is a python based framework (that is under the hood an abstraction of React.js) that is especially targeted at data-driven applications as it includes data visualization possibilities via plotly, enriched by leveraging bootstrap UI components [118].

**Persistence:** Data is stored persistently in one central PostgreSQL [119] database which has been designed for storing domain aggregates rather than relational dependencies. Event data and survey data are provided via csv files whereas an initial stack of product change suggestions is stored in a json format (as seed files).

**Testing**: Unit and integration tests (in total 175) are conducted with pytest [122] (python). A dedicated testing database (PostgreSQL) is also part of the test suite.

**Design:** UML Models are designed using plantUML [126], which allows for code-compiled UML models in various categories (use case, class diagrams, ...). In cases that exceeded plantUML's capabilities, draw.io [120] has been used.

**Environment:** Docker [121] (with docker-compose) is used to spin up containerized Postgres databases (also for testing). Makefiles [127] are used for easing command-line execution of Frontend, Backend, and Tracking Services. Dependencies are installed in python virtual environments (virtualenv [128]), so isolated systems are created which empowers easier installation of HypGen on other systems.

A technical extension for evaluation purposes should also be mentioned, although it is not relevant to HypGen's core tech stack. An external telemetry service (for evaluation of the DSR artifact) that records mouse locations and selected user interaction events is implemented in a custom fastapi app. Callbacks in the frontend (written in python) send requests to store interaction events to the telemetry service. For recording mouse movements, a custom javascript snippet is developed which enriches the Dash application. Persistent data storage for evaluation telemetry data is achieved in a dockerized PostgresSQL database as well.

The development of the software followed the general principles found in Domain-Driven Design (DDD) [129] and was inspired by a public 'pythonic' DDD implementation [130]. Mapping concepts in the course of hypotheses generation to Bounded Context (BC)s allowed to slice the software into a way that resembles a modular monolith architecture. This software architectural design can be understood as a way to achieve modular design,

but not at the cost of introducing too much technical complexity with the need to manage multiple microservices [131]. The different modules are closed entities that communicate with RESTful API calls with each other. While leveraging the same database, all tables have clear module ownership and are not joined nor queried from another module. In the following, when BCs are referred to, the reader can translate it to its technical representation in the form of a module.

### 6.2.2. Scenarios

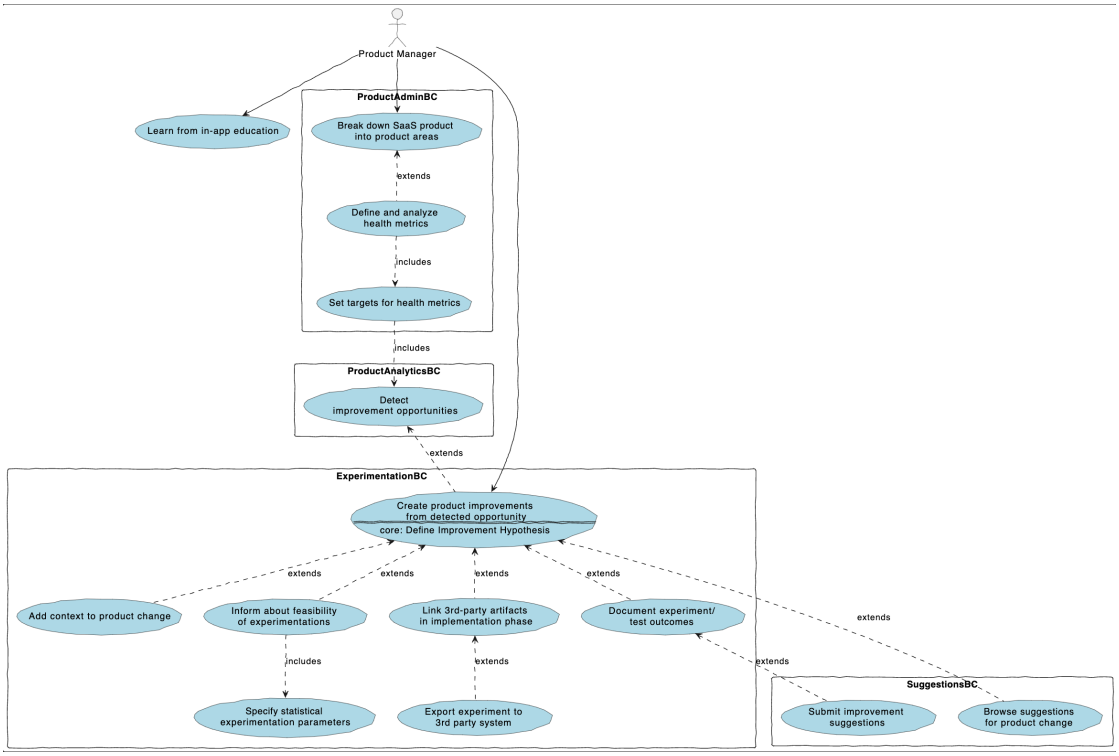



Figure 6.17.: Use Case Diagram

In the following *UML use case* diagram, the core use cases of HypGen are illustrated (cf. Figure 6.17). In general, the DSS is tailored for Product Managers so this persona acts as the user of the use case diagram. The use cases are separately listed but with their extend/include dependencies as well as their primary mapping to BCs. In the *ProductAdmin* BC, HypGen offers use cases to break down an entire SaaS application into logical parts and to define product health analysis for it (incl. desired target values to spot gaps). Further, the *Product Analytics* BC offers the use case to detect potential improvement opportunities.

The BC *Experimentation* with the biggest use case offering then allows to define the product improvement with the improvement hypothesis in its core. Various use cases

that aim to improve the quality of the hypothesis extend this primary use case. E.g. use cases are offered that let the Product Manager add more context to a product change (qualitative user research, strategic insights), and inform about the feasibility of different experimentation set-ups (e.g. Pre-Post test vs A/B test). Further, it has functionality to cover the use cases to link third party tools (e.g. jira tickets, github pull requests) for traceability. As a key functionality, it allows to create randomized controlled experiments with the specification from HypGen in the experimentation software PostHog. Ultimately, it also allows to document the final result of the hypothesis test's outcome and the implication for (not) rolling out the desired product improvement.

As a helping hand, the BC for *Suggestions* offers two use cases that support the definition of the improvement hypothesis and to close the long-term learning loop of experimentation. First, the use case to browse suggestions for a product change is implemented which primarily builds on published A/B test results and their changes or even industry best practices. Second, to build up experimentation intelligence, *submit improvement suggestions* allows the Product Manager to store an experiment's outcome as a learning which will enrich future suggestions for Product Managers - once they work on similar topics.

As a generic use case, accompanying the Product Manager in her learning journey has been illustrated as well which is supported by manifold ways like tooltips, extensive contextual views, and an intuitive workflow.

### 6.2.3. Logical View

In the following, the author presents core aspects of the solution which can be best described with a selection of class diagrams.

#### Layered DDD Architecture

While HypGen is designed in a modular monolith manner, i.e. separated horizontally into four BCs[7] (*Product Admin*, *Product Analytics*, *Experimentation*, *Suggestions*), the solution is also layered vertically (cf. Figure 6.18). The visualization portrays the usual design which is applied in all BCs. It's worth noting that there exist some corner cases which are briefly illustrated after the general case.

In general, the user interface handles visualization of data it receives communicating via the *Presentation Layer*'s RESTful APIs. The API in the presentation layer delegates the logic to the *ApplicationLayer*. In specific, it delegates to Query or Command *UseCase* implementations, a concept influenced by CQRS design [132]. To get the job done, the use cases have a *RepositoryImplementation* which allows to find or create/update entire *DomainAggregates*. In order to execute business logic, the *UseCases* delegates this to functionality of the *DomainModelLayer*, and in specific to the domain's *AggregateRoot*. All domains are modelled to have one central *DomainAggregateRoot* which ensures consistency of further *Entities* or *ValueObjects*. The *DomainAggregateRootRepository* acts

[7] Excluding the SharedKernel in this context

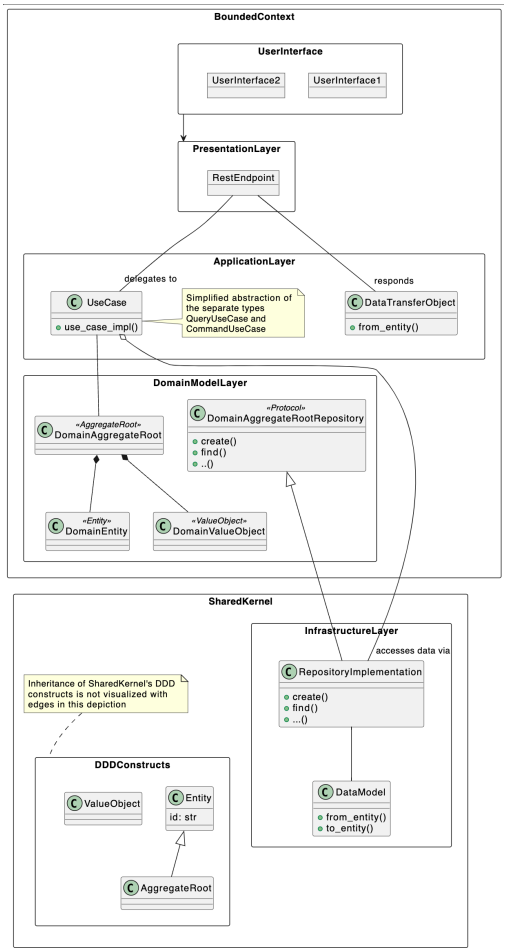



Figure 6.18.: Class Diagram: Layered Architecture Kernel

as an interface that the *RepositoryImplementation* (in *SharedKernel*) must implement. The *Repositories* are designed to execute CRUD actions on whole *DomainAggregates*, not just parts of it. The implementation details of *DDDConstructs* such as *Entities*, *ValueObjects*, and *AggregateRoot* (as well as generating string ids) are delegated to the *SharedKernel*. Inheritance is used to benefit from the desired constraints.

For complex objects, *Domain Factories* are used - e.g. when creating complex pipelines in the BC *ProductAnalytics*. Also, *Domain Services* are used, e.g. when evaluating multiple validation scenarios in the *Experimentation* BC. However, mostly objects are created via classmethods of the *AggregateRoot*, and logic is mostly handled in the *AggregateRoot* or in parts in the *UseCase*, not in a *DomainService*.

Ultimately, the *InfrastructureLayer* is situated in the *SharedKernel* which handles the logic for the persistence layer. The *RepositoryImplementation* implements functionality that is defined in the *DomainAggregateRootRepository*. For querying or updating, the *Repository* usually looks for the corresponding *DataModel*, converts it, and returns it in

the form of the *Domain Aggregate*. For storing, it receives the whole *DomainAggregate*, converts it to the *DataModel*, and stores it persistently in the database.

### Shared Kernel

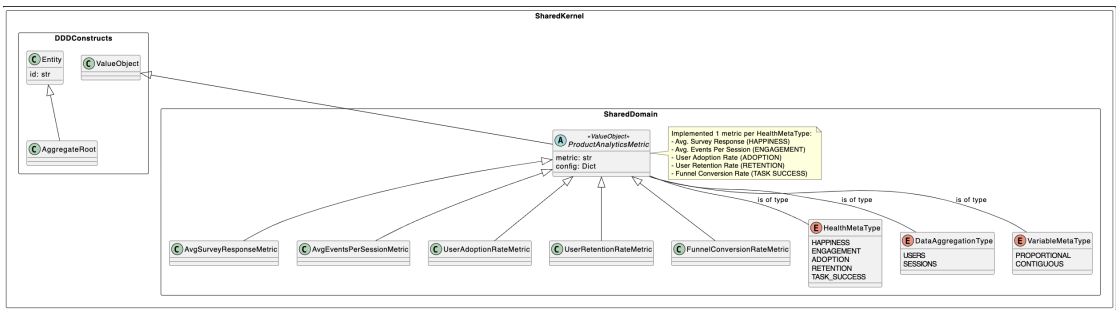



Figure 6.19.: Class Diagram: Shared Kernel

The *SharedKernel* holds the core generic functionality of the whole system which should be reused by its other parts (cf. Figure 6.19). Most importantly, it consists of two groups of logic. First, it implements DDD's [129] specialties, i.e. *Entity*, *Value Object*, and *Aggregate Root*. Doing so, the *SharedKernel* controls centrally e.g. how identity is handled in *Entities* vs. *Value Objects*. Second, a lot of logic in the whole system is concerned with metrics, either to analyze or to optimize them. To have a joint understanding of metrics as well as their specialties, they are part of the *SharedKernel* and must be reused by other parts of the software. For instance, it describes which metrics are available, which HEART categories they belong to (*HealthMetaType*), and what kind of metrics they are (*VariableMetaType*).

### Product Admin

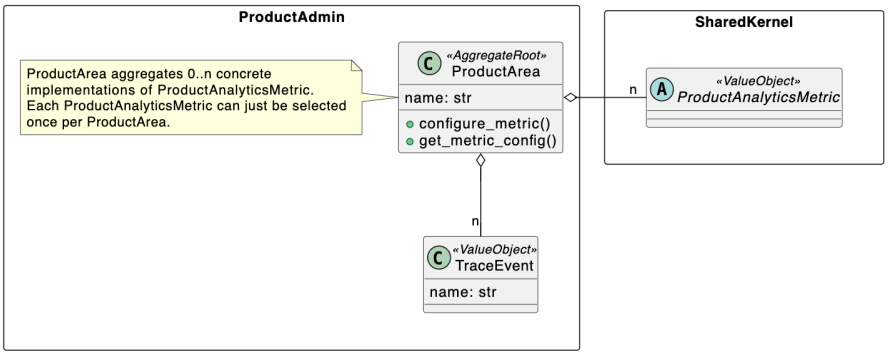



Figure 6.20.: Class Diagram: Product Admin

As seen above, Product Managers can define their logical part of the software product (i.e. a product area) and analyze its health in the *ProductAdmin* BC (cf. Figure 6.20). With this respect, the *ProductArea* is the *AggregateRoot* which includes a list of *TraceEvents* (the events that the user selects). Further, the *ProductArea* controls setting up a set of *ProductAnalyticsMetrics* and also controls how the settings are returned back once set up. The *ProductArea* aggregate deals with the creation and configuration of the logical sub-component of the SaaS product.

### Product Analytics

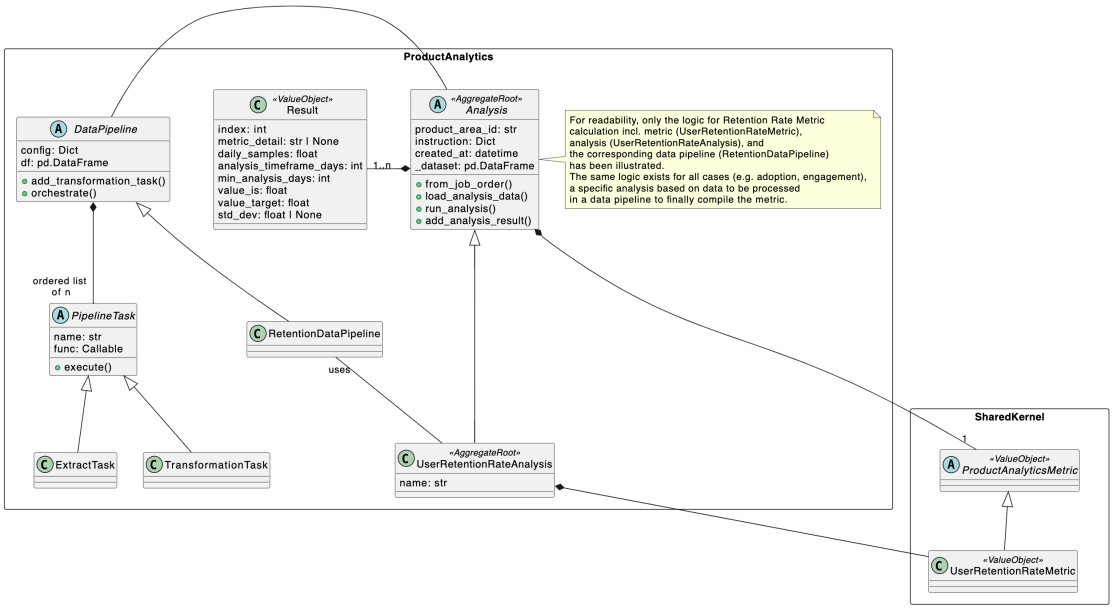



Figure 6.21.: Class Diagram: Product Analytics

The *ProductAnalytics* BC (cf. Figure 6.21) has the purpose of analyzing product health metrics for a dedicated *ProductArea* in order to identify improvement opportunities. The *AggregateRoot* is an *Analysis* that has different concrete implementations, one for each metric to analyze. E.g. if the metric *UserRetentionRateMetric* is the target of the analysis, then the AggregateRoot is a *UserRetentionRateAnalysis*. The *Analysis* stores an ID that relates it to the *ProductArea* for reference. Further, it stores some instructions to compile the metric which it receives via a *JobOrder* (not part of the depiction). The *JobOrder* basically is an instruction authored by the *ProductAdmin* BC of which properties need to be considered for the analysis (e.g. timeframes, target rates, ...). After the *Analysis* is set up, it loads the data necessary for the *Analysis* (custom implementation for each *Analysis* class), runs the *Analysis* (custom implementation of the *run_ analysis()* method for each *Analysis* Class), and adds the analysis results.

As analyzing data does not solely consist of reading a csv file, but executing a set of

well-thought-through data transformations and cleanings, each *Analysis* makes use of a dedicated *DataPipeline* to prepare the dataset needed to compile a certain metric. A *DataPipeline* again is an ordered collection of data extraction and transformation tasks (*PipelineTask*) that each have a name illustrating its purpose and have a *Callable*. A *Callable* is a function that takes in an existing *DataFrame* (except for the *ExtractTask*) and some configuration to return the processed *DataFrame* again. Concatenating multiple *PipelineTasks* then basically makes up the sequential *DataPipeline* where a *DataFrame* ready for analysis is incrementally transformed. The analysis' *Results* are detailed in nature and store beyond the metric's value identified meta-information such as target value, daily sample sizes, and standard deviations.

### Product Experimentation

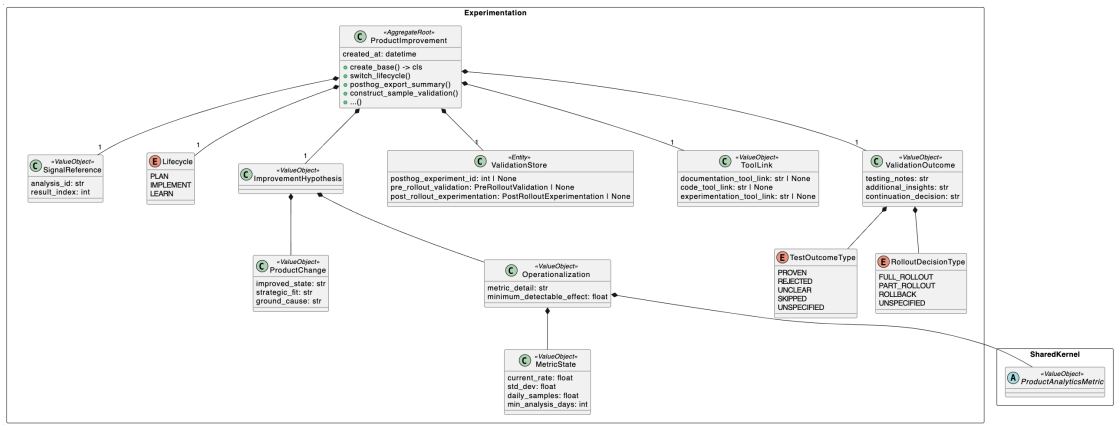



Figure 6.22.: Class Diagram: Experimentation

The *Experimentation* BC (cf. Figure 6.22) has the purpose of specifying a *ProductImprovement* which is defined as a statistically testable hypothesis that is wrapped into an *Experimental* setting for validating a potential impact. The *ProductImprovement* is initially created using the *create_ base()* method, i.e. a base *ProductImprovement* can be created from an *ImprovementOpportunity* (which is the result of an analysis in the *ProductAnalytics* BC). For this purpose, also a reference to the opportunity is part of the *ProductImprovement* which allows the frontend (or any other logic) to request more information about the initial opportunity. The *ProductImprovement* has a *Lifecycle* that corresponds to either *Plan* (general concept incl. hypothesis and experiment planning), *Implement* (coding the feature, collecting usage data for hypothesis testing, ...), to *Learn* (deriving learnings from experiment results). Further, as a core element, the *ProductImprovement* includes an *ImprovementHypothesis*. It is comprised of the contextual element, the *ProductChange*, which describes changes in the software as well as why it is solving both a problem for the user and makes sense for the business. In addition, it also includes an *Operationalization* of the hypothesis, basically which target metric

should be improved - and by how far[8]. In this case, the *SharedKernel*'s definition of a *ProductAnalyticsMetric* is used again. The historical values of the data analysis are also stored in the *MetricState* which serves as important input for sample size and duration estimations for the subsequent experimentation. Continuing, the *ProductImprovement* includes a *ValidationStore* (more details, see below) which stores information on selected pre-rollout validations (to make an *ImprovementHypothesis* more robust) and selected post-rollout experimentation (to test the validity of the *Hypothesis* with real usage data). For traceability reasons, the *ProductImprovement* also includes a *ToolLink*, which is a collection of links to external tools where implementation takes place. Ultimately, the *ProductImprovement* includes a *ValidationOutcome*, which stores the hypothesis test's *ValidationOutcome* that also documents the corresponding rollout decision and further learnings.

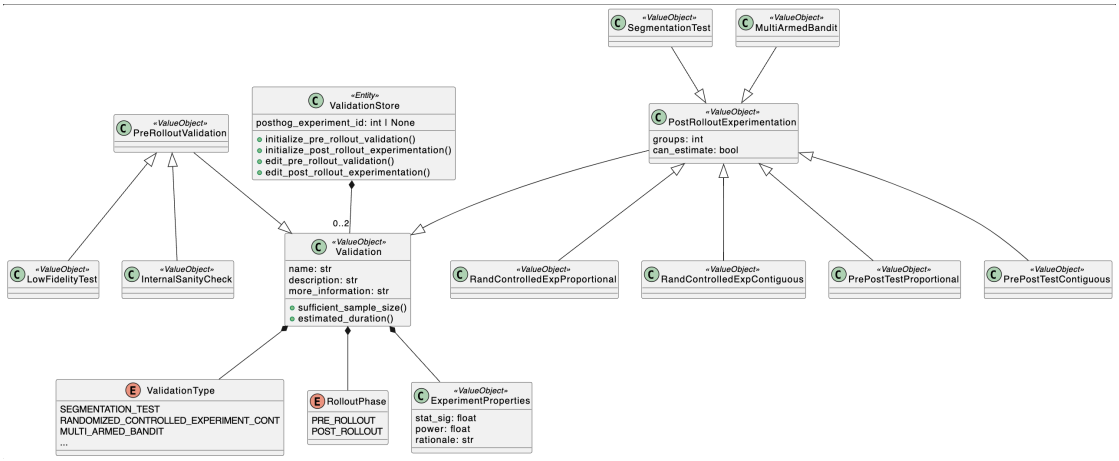



Figure 6.23.: Class Diagram: Validation Store Detail

The *ValidationStore* (cf. Figure 6.23) acts as an entity to manage (initialize/edit) the selection of *Validations* and *Experimentations*. Besides a *posthog_ experiment_ id* (external resource identifier to the experimentation system), it stores up to 2 *Validations*. A *Validation* can be understood as an abstract concept whereby only its concrete implementations play a role. Each *Validation* has a name, description, and more_information text which serves as in-app education material to be visualized in the UI. Where implemented for *Validation*'s concrete implementations, also estimations for sample size[9] and experiment duration can be conducted for a certain *Validation*. Supporting sample size estimations, *ExperimentProperties* (statistical significance, statistical power, ...) are also part of each *Validation*. The *ValidationStore* can hold up to one instance of either *PreRolloutValidation* and up to one instance of *PostRolloutExperimentation*[10].

[8] As was described earlier, having additional guardrail metrics to control for unwanted effects of a product change is seen as a good practice in CEx. However, to keep the scope reasonable for HypGen, the option to specify guardrail metrics has been omitted.

[9] by leveraging publicly shared algorithms (cf. [133, 134])

[10] Supporting up to one of each kind is seen as reasonable for the scope of the thesis. However, in theory

A *Validation* is either a *PreRolloutValidation* or a *PostRolloutExperimentation*. A *PreRolloutValidation* specifies *Validation* opportunities one can undertake before the rollout of a feature, e.g. a *LowFidelityTest* (testing a visual prototype) or *InternalSanityCheck* (e.g. with customer-facing colleagues). A *PostRolloutExperimentation* specifies experimentation settings to test the potential product improvement with real users after rollout, with one of many possible experiment settings. Not all *PostRolloutExperimentations* have the same amount of groups (e.g. in a *RandomizedControlledExperiment*, 2 groups are assumed), nor are all of them implementing sample size calculations (e.g. *Multi-Armed Bandit*). However, each *PostRolloutExperimentation* where sample size estimations are generally feasible without introducing major complexity, i.e. those that are marked as (*can_estimate=True*), can deliver sample size and time to results (*estimated_duration()*) estimations.

### Suggestions

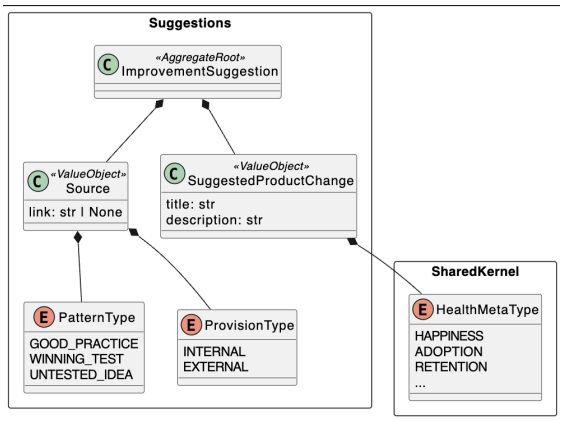



Figure 6.24.: Class Diagram: Suggestions

The *Suggestions* BC (cf. Figure 6.24) acts as a stack for potential ideas to further improve certain product health categories (*HealthMetaType*). The *AggregateRoot* is the *ImprovementSuggestion* which has a *SuggestedProductChange* and a *Source*. The *SuggestedProductChange* basically holds what should be done (title, description) to drive metrics in a certain health dimension (*HealthMetaType*) according to the suggestion. Further, the *Source* includes what kind of suggestion this might be. It might be provisioned internally (other Product Managers of the same company) or externally (interesting features from competitors, ..). The quality of the suggestion is controlled for in the *PatternType*, which can be a *GoodPractice* (common agreed tricks/patterns in the industry), a *WinningTest* (e.g. a proven hypothesis that has worked out in the past), or an untested idea (without any further backing). The BC is rather simplistic and is more concerned with storing (and validating) state than executing complex business logic.

nothing speaks against having multiple pre-validations

### 6.2.4. Process View

#### General Communication Logic

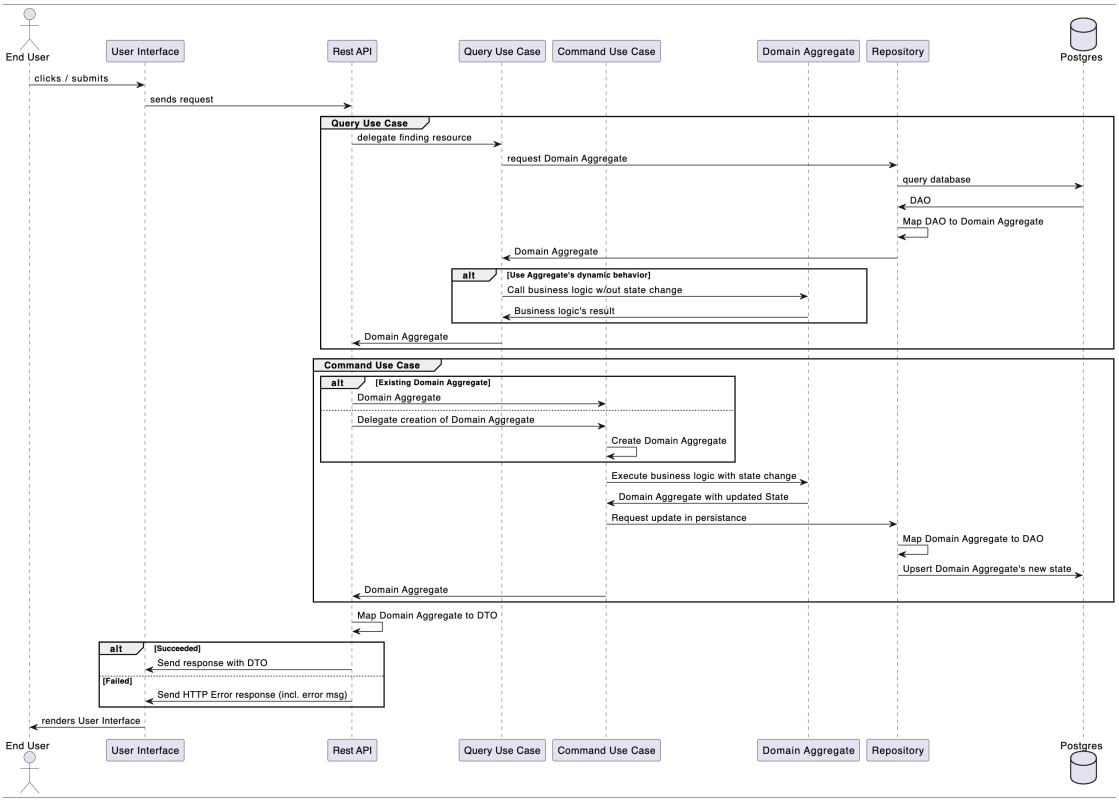



Figure 6.25.: Sequence Diagram - General Communication

First, the author presents the usual logic regarding how the communication flow works (cf. Figure 6.25). In the sequence diagram, a combination of *Query Use Case* and *Command Use Case* is presented. This is the case when an existing resource is first found and then updated (e.g. *PATCH* or *PUT* request). When an entirely new *Domain Aggregate* is created, the Query Use Case can be skipped (e.g. *POST* request). When only existing Domain Aggregates are queried (e.g. via a *GET* request), the Command Use Case can be skipped.

Assuming we have to do with a Query and Command Use case, it starts when the end user is clicking on a page / UI element or submits a form (*User Interface*). This triggers a request to the Backend's *REST API*. The *REST API* (or more explicitly its controller) is delegating finding the resource to the Query Use Case. This one in turn further delegates it to the *Repository* which accesses the data from the database (marked as *Postgres*). Having mapped the *Data Access Object (DAO)* to the Domain Aggregate, the repository is returning the *Domain Model* back to the Query Use Case. Once business logic without a state change is involved (no update in database necessary), the Query

Use Case is delegating it as well to the Domain Aggregate. A real-life example would be the feasibility check for a given hypothesis and experimentation combination. Finally, the Domain Aggregate is returned back to the controller.

Now to make a change to an existing Domain Aggregate, the Domain Aggregate is handed over to the corresponding *Command Use Case* functionality. Optionally, once it involves creating a new Domain Aggregate, this would also be handled in the *Command Use Case*. In this case, obviously, no existing Domain Aggregate would be handed over. Then, business logic to change the state is delegated to the Domain Aggregate which ensures consistency. The updated Domain Aggregate is then handed over to the *Repository* that initiates the update (or creation) of the *DAO* in the database (Postgres). Ultimately, the Domain Aggregate is returned to the *REST API*.

During all this, once an error occurs, the *REST API* responds with a *HTTP Error*, else it sends an endpoint-specific *DTO (Data Transfer Object; json format)* which is created based on the *Domain Aggregate*.

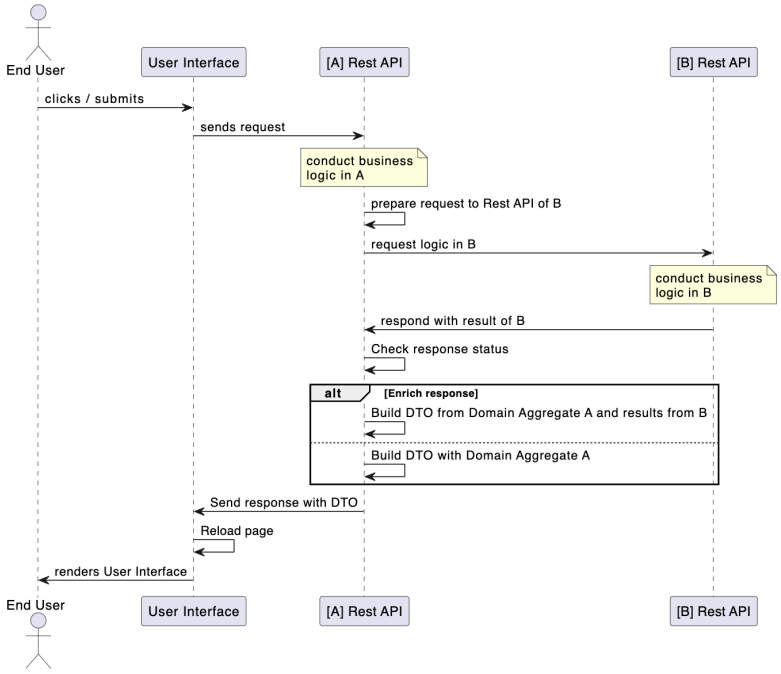



Figure 6.26.: Sequence Diagram - Cross Bounded Context Communication

To add a further important aspect, communication between two BCs is illustrated (cf. Figure 6.26). Sometimes, a BC needs some functionality (or the current state of data) of another BC. Communication between BCs takes place via REST API calls (which in turn delegate logic further like presented above). Once the user requests functionality in the *User Interface* which spans multiple BCs, usually it has a primary source (e.g., in this case, BC A), so the request is sent to *[A] REST API*. After receiving the request, the BC then executes business logic and returns a *Domain Aggregate*.

The *Domain Aggregate*'s information (such as references to a resource of another BC) is used to construct another REST API call to the other BC's *[B] REST API*. B then returns the result which (optionally) enriches the desired result in A. One example is that a *ProductImprovement* holds the reference to another BC's *AnalysisResult*, which can then be queried further to get details such as to which *ProductArea* it belongs. The result must not be necessarily enriched, it could also be that A's API just checks if the delegation has been successful and returns a generic response indicating the operation's success.

As always, exceptions are present. There also exist cases where the UI first fetches BC A's request and evaluates its result. Only after that, it requests another BC's API. This is a case where more logic is happening in the UI which is omitted here as it is a quite straightforward case.

### 6.2.5. Development View

From a development perspective, the component diagram (cf. Figure 6.27) illustrates well the structural relationships and dependencies between HypGen's system components. In this case, the *UML Component Diagram* is modeling the four core subsystems. The interfaces of these systems can be understood as the main (aggregated) use cases that each of the subsystem's RESTful APIs are providing. The shared kernel, external systems as well as persistence and data storage is excluded from this depiction. This allows for a greater focus on these subsystems that are delivering the core value for hypotheses generation.

The *ProductAdmin* subsystem provides interfaces to manage a product area and specify health metrics using Google's HEART framework. This is delegated to the *ProductArea* component which is an aggregation of classes to manage a sub-part of a software system. It provides an interface to query a single product area or multiple product areas.

The *ProductAnalytics* subsystem provides the interfaces to get *Ranked Opportunities*, i.e. product improvement opportunities which are the result of analyzing specific health metrics. As the *ProductAdmin* subsystem manages the *ProductArea* and its corresponding product health specifications, it requires the *AnalyzeHealthMetric* interface to analyze the data. In order to spot opportunities to improve a product area, a *DataPipeline* component is providing ETL orchestration, e.g. a set of transformation tasks that analyze user retention based on product data.

The *Experimentation* subsystem provides the interfaces to manage a *ProductImprovement*, i.e. the extended hypothesis of what to improve in the product and how to validate if it was successful. For purposes of validating a hypothesis, an interface for an *Experimentation Check* is provided which delivers educational content about each experimentation type as well as feasibility estimations (sample size, estimated time to results). Additionally, the interface *Export Experiment* offers the possibility to send the experiment to a third-party tool. In order to create a *ProductImprovement*, the subsystem requires an improvement opportunity which is provided by the *ProductAnalytics* subsystem.

The *Suggestions* subsystem provides the interfaces to retrieve *Hypothesis Inspiration* once working on brainstorming a potential product change. This interface suggests

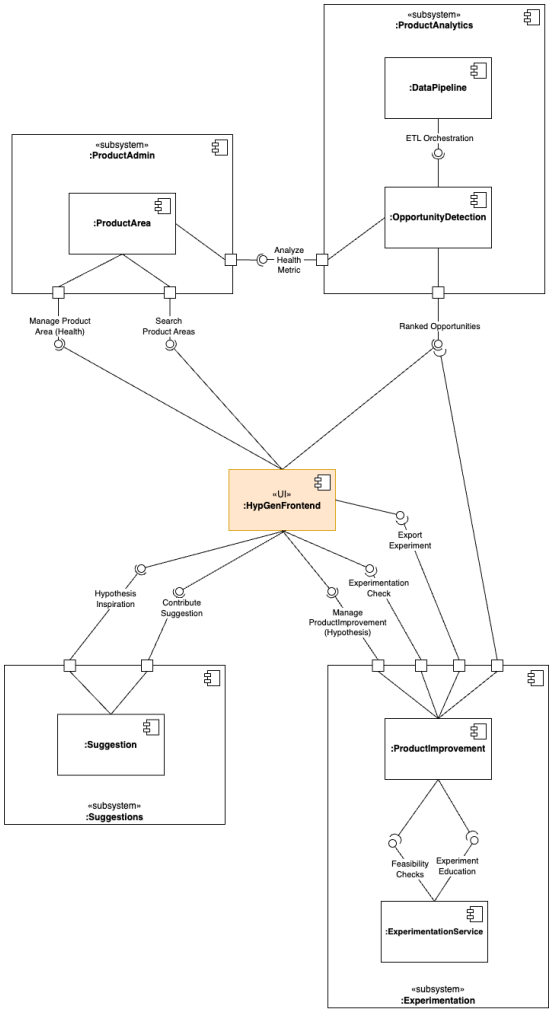



Figure 6.27.: Component Diagram

potential solutions given a certain product health category to improve. In addition, the subsystem provides the interface to also store a suggested product change (*Contribute Suggestion*).

The *HypGenFrontend* requires a variety of interfaces to power its DSS. The user interface leverages the relevant capabilities of each subsystem once needed.

### 6.2.6. Deployment View

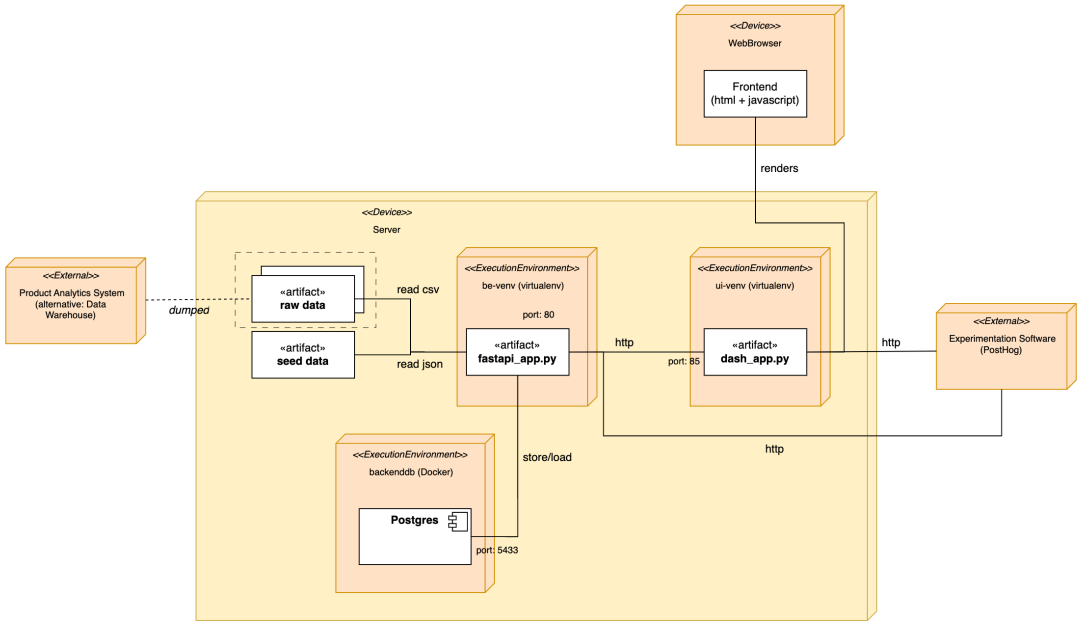



Figure 6.28.: Deployment Diagram

The physical view is illustrated with a *UML Deployment Diagram* (cf. Figure 6.28). It depicts HypGen as well as external systems which it has interactions with.

Starting from the top, the end user interacts with her *WebBrowser* where HypGen's frontend is rendered. While dash is python-based, it compiles to React.js under the hood to render the UI in the web browser.

The dash app which includes the frontend code is deployed in a python virtual environment where the script to host the app (*dash_app.py*) is run. The dash app is accessible on *port 85*.

The dash app communicates with http requests with the backend which is also executed on a server device. Again, a virtual environment is used where the main *fastapi_app.py* file is run with uvicorn to serve requests via *port 80*.

To access data, two options are worth highlighting. First, storage-optimized representations of the domain aggregates are stored in a *Postgres* database which lives in a *Docker* container and is accessible via *port 5433*. Each domain aggregate has its own table in the database. Second, raw data files are stored on the server which are loaded for calculating metrics (e.g. events.csv, surveys.csv). Also, seed files for the suggestion functionality are stored there.

For external communication, two nodes are visualized. First, the link from a *Product Analytics System* to the *raw data* is depicted. It describes the relation between how

HypGen receives real product usage events and survey data. To stay within the scope of the thesis project, no active communication has been implemented, but the data was dumped and copied to the server. Second, the backend communicates when creating an experiment with an external *Experimentation Software* (web application) called Posthog. While the communication edge for creating the PostHog experiment from HypGen lives in the backend, fetching the current state of the PostHog experiment (get request) lives in the dash app (frontend) directly as no complex logic is involved.

# 7. Evaluation

This chapter deals with the final evaluation of the artifact which was conducted with considerable scientific rigor. The methodology of the evaluation and its results are discussed in depth. Finally, the results are summarized to answer whether the artifact can be considered as improving hypotheses generation in the context of CEx.

## 7.1. Method

Evaluation in DSR helps to assess the relevance of the developed artifact. A DSR artifact is 'judged both by the fitness of the designed artifact as a solution in the application environment and by the level of new research contributions made to the appropriate knowledge bases' [135]. For this thesis DSR project, it means that the evaluation needs to assess if HypGen solves the identified problems in the interview study and addresses the research gap in hypotheses generation in the context of CEx. Further, it should yield more nuanced insights so that future design artifacts can benefit from the design knowledge generated as part of this DSR project. So, this evaluation will ensure that both the initial research question of the whole DSR project as well as the degree to which problems are solved with HypGen will be answered. Moreover, the evaluation's analysis should ensure with a wide variety of different analysis approaches that extensive knowledge is generated.

### 7.1.1. Evaluation Strategy

The overall research question that motivated the development and evaluation of HypGen is the following: *'How should a Decision Support System be designed to improve the generation of testable and valuable product improvement hypotheses in a SaaS startup context'*? The relevance of this question has been emphasized by synthesizing insights that were the result of the qualitative interview study. While interviewees at the SaaS startup report that they are making a lot of assumptions and are hypothesizing, they are neither formalizing their hypotheses nor do they test their ex ante hypothesis after rollout with scientific rigor. And, it became apparent that not only the definition of statistically testable hypotheses is of high interest, but also to ensure that hypotheses target at solving valuable problems while testing probably worthwhile product changes.

According to Peffers et al. [136], the designed research artifact HypGen can most likely be characterized as an 'instantiation' which is evaluated in a mix of 'prototype' and 'case study' setting [136]. Other authors may frame such an evaluation as a 'pilot study with a sample of targeted end users' [137]. HypGen is evaluated in a naturalistic approach as opposed to artificial, i.e. real people use real technical systems to conduct real tasks

to solve real problems [138]. In this evaluation, HypGen is loading actual usage events and survey data of the SaaS product that the subjects are working on in their day-to-day jobs. Hence, subjects are trying to come up with hypotheses to improve or extend already existing functionality of their real-world software product.

While two intermediate evaluations have taken place with qualitative improvement feedback (which was already described in the previous chapter), the ex post evaluation assesses the final version of HypGen. The now following summative evaluation is well suited to assess the results of finished development and to reason about the overall project's success [139].

### 7.1.2. Evaluation Criteria

#### Overall Research Evaluation

To control how far the overall artifact's core research question has been addressed by HypGen, subjects have been asked two pairs of Likert-Scale and qualitative follow-up questions. Subjects are asked to what degree HypGen:

- supports that product improvements are **more likely translated into statistically testable** hypotheses
- helps with the generation of **valuable** hypotheses, i.e. hypotheses that probably include effective product changes that solve impactful improvement opportunities

The results of these two questions together can yield an overall view if the research question was answered successfully, i.e. if HypGen can be understood as a design to improve hypotheses generation in the case study setting.

#### Artifact's Evaluation Criteria

Going beyond a high-level examination of answering the research question, DSR evaluation criteria are aimed at assessing the fitness of the presented artifact to solve problems in the application domain. The careful selection of effective evaluation criteria is ensured by first reviewing influential authors in the DSR community. Prat et al. both show that common evaluation criteria exist in DSR literature to get inspiration from, but that a good set of evaluation criteria is always dependent on the exact setting [140]. Reviewing criticisms on the DSR project landscape which diagnose that DSR projects concentrate too heavily on utilitarian goals (e.g. simplicity, usefulness) [135] and tend to focus less on the organizational context of its application [140], the author broadens the scope of evaluation criteria. By including additional categories such as *Business Alignment*, *Applicability* (in other domains), or *IT Landscape Fit*, the research aims to evaluate the artifact beyond utilitarian goals. The author lists the eight finally selected evaluation criteria as well as indications of how data to assess each category is collected (cf. Table 7.1). All survey questions that gauge subjects' opinions are combined scalar and qualitative follow-up questions where subjects have been prompted to explain why

they chose a certain score. All evaluation criteria map 1-to-1 in answering the Goodness Criteria of the design objectives that have been described previously (cf. section 4.4).

| Criterion | Description | Evaluation |
|---|---|---|
| Accuracy | Does the tool deliver the correct health metrics as well as sample size calculations given certain input parameters? | Unit Tests |
| Applicability | Is HyGen perceived as applicable in settings other than the case study (i.e. at other SaaS companies)? | NPS question |
| Business Alignment | Does the tool have the potential to contribute to greater adoption of CEx? | 5pt Likert-Scale |
| Completeness | Does the tool offer features that solve the majority of relevant tasks in the course of defining product improvement hypotheses? | 5pt Likert-Scale |
| Fit into IT Landscape | Does the tool fit into the suite of tools Product Managers are already using? | 5pt Likert-Scale |
| Learnability | Does the tool make it easy for users to learn how hypotheses generation with HypGen works? | 5pt Likert-Scale |
| Simplicity | Is the tool easy and intuitive to understand and to get desired actions done? | 5pt Likert-Scale |
| Traceability | Does the tool help to connect activities of the origins of the product improvement hypothesis until its release and effective outcome? | 5pt Likert-Scale |
| Usefulness | Is the tool perceived as useful in the course of defining valuable, testable product hypotheses? | 5pt Likert-Scale |

Table 7.1.: Artifact's Evaluation Criteria

### 7.1.3. Execution

#### Subjects

The 5 evaluation subjects are all working in a core product management role at the company in scope. Similar to the interview pre-study, the subjects consist of 4 Product Managers and 1 UX Researcher. Comparable DSR studies leverage a similar number of evaluators (cf. 7 subjects in a study on a DSS in clinical environment [141]). Due to changes in the company, 2 Product Managers (S3, S4) from the interview study have been replaced with 2 new Product Managers (S6, S7) in the evaluation that are with the company at the time of the evaluation[1]. Below (cf. Table 7.2) is a table of evaluation subjects and a note if they have been part of the initial pre-study or intermediate prototype (feedback) evaluations[2].

[1]The evaluation takes place over a year after the pre-study interviews

[2]Subject S4 who has participated in the interview study has also been part of intermediate evaluations, but was not available anymore for the summative evaluation

| | Role | Pre-Study | Intermediate Evaluation |
|---|---|---|---|
| S1 | Product Manager | yes | no |
| S2 | Product Manager | yes | no |
| S5 | UX Researcher | yes | yes |
| S6 | Product Manager | no | yes |
| S7 | Product Manager | no | no |

Table 7.2.: Evaluation Subjects

### Setting

The researcher set up individual appointments with the subjects at their workplace. All appointments took place in June or July 2023 and lasted about 1h each. The evaluation sessions took place in closed meeting rooms where the subject and the researcher conducted the evaluation in a 1-on-1 setting. At the start of the evaluation meeting, the researcher sent a digital questionnaire to the subject which was opened at a second screen. After completing ex ante questions to find out about experience in hypotheses generation and experimentation, the subject received a notebook on which HypGen was installed and running. The second screen was available during the evaluation. The screen was used by subjects to read the task they were prompted to complete at any time and allowed them to switch to the next task. During the evaluation, subjects were motivated to think aloud when completing tasks with HypGen. After the completion of all tasks, the post hoc part of the questionnaire continued and the subjects were asked to provide answers based on their experience with HypGen. The post hoc part is meant to both measure evaluation criteria and to find out more nuanced aspects of subjects' perception of the artifact.

From a technical perspective, the notebook that the subjects used had extensive mouse location recording and user interaction tracking in place. Further, the notebook had an internet connection and once subjects would feel the urge to go on other websites during the evaluation they could do so[3]. To record the general usage flow of each subject as well as to review what they mentioned in their think-aloud, the screen has been recorded including audio.

### Tasks

Evaluation subjects were prompted to conduct a sequence of five tasks. The five tasks all build up on each other and allow subjects to experience the full scope of HypGen. The tasks begin with setting up a product area and ultimately end with documenting the experiment outcome for a generated hypothesis that was meant to improve the initially created product area. After having answered ex ante questions, subjects were presented with a textual task prompt for the first task. Once having completed a task, interviewees

[3]No subject tried to research information or similar on the web

continued to the next task prompt. The task prompts stood for themselves and were meant to be self-explanatory. Upon request, the author was available for possible queries and clarifications. Subjects have been prompted with the five sequential tasks below:

**Task 1 - Create a Product Area incl. Tracking Events:** Create a product area that describes a logical part of your software product. Use event tracking to capture relevant user interactions as part of your product area.

**Task 2 - Specify Product Health Metrics:** Leveraging the product area you have just created - configure product health analysis (HEART) for 2+ health dimensions and run the analyses.

**Task 3 - Review and Select Improvement Opportunities:** Review the improvement opportunities for your product area which have been detected by HypGen. Select a promising opportunity to improve your product area.

**Task 4 - Define an Improvement Hypothesis & Decide For a Feasible Experimentation Set-Up:** Conceptualize the different parts of your product improvement hypothesis and evaluate different experimentation approaches to test your hypothesis after release. Note: For inspiration and evaluating a hypothesis' feasibility, you can leverage the software's capabilities.

**Task 5 - Define the Hypothesis Outcome:** Simulate further steps of your product improvement during the lifecycle from Plan > Implement > Learn. The latter being already after the release. In the learn phase, make up a theoretical outcome for your product improvement's validated hypothesis and save it.

### 7.1.4. Analysis

The author analyzes the evaluation's results in a variety of ways - combining survey responses with quantitative data and qualitative observations. Survey responses are evaluated in a quantitative manner and presented in greater detail than usual to focus on individual observations as well (e.g. showing individual values in graphs and plots, not just averages). In addition, qualitative follow-up questions are drawn upon to increase the meaningfulness of quantitative answers. Also, additional questions than solely to measure the evaluation criteria are part of the questionnaire to increase the degree of knowledge generated. Moreover, interactions with different AoIs (i.e. logical groups of user interactions) have been visualized to learn about task completion behavior in a timeline perspective, a method which is often found in user studies with few subjects but high analysis depth (cf. [142]). This angle is enriched by mouse heat maps which are used wherever suitable to further understand how subjects are using HypGen to achieve their tasks. Third, screen recordings and corresponding think-aloud audio recordings are reviewed to enrich the overall understanding. These qualitative observations are used whilst e.g. interpreting time to task completion (from reading out the task to completing it) or whilst enriching other types of analyses with observational data. Information from think-aloud is rather selectively drawn upon to bring nuance to other types of analysis methods as think-aloud did not offer larger standalone value.

## 7.2. Results

### 7.2.1. Ex Ante Insights

#### Experience in Hypotheses Generation and Experimentation

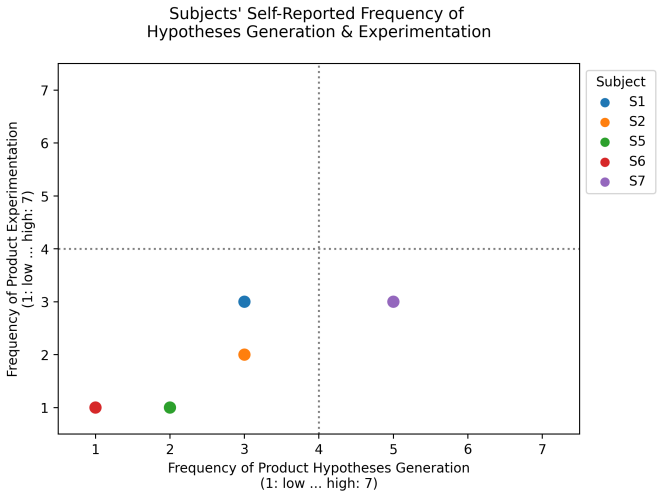



Figure 7.1.: Subjects' Self-Reported Application Frequencies

To understand the current maturity of subjects working with product hypotheses and experimentation practices, the author asks two questions before demonstrating the prototype. The first question (Q1[4]) gauges how often subjects define product changes as testable hypotheses. Another question (Q2) asks how often they are testing these hypotheses after rollout in experimentation settings. For simplicity, answers to Q1 are displayed on the x-axis, and answers to Q2 on the y-axis.

Results show that subjects consistently classify themselves as rather seldom defining product hypotheses as well as conducting experiments to test these hypotheses (cf. Figure 7.1). 4/5 subjects can be found in the bottom-left quadrant with the remaining subject nearby - all indicating a rather low frequency of applying product hypotheses generation or experimentation.

Supporting this, respondents were asked a follow-up question (Q3) on how far they have ever applied a selection of experimentation approaches. One subject has never applied post-rollout experimentation, and others have used some experimentation approaches. Only 2/5 subjects have ever conducted a randomized controlled experiment (e.g. A/B test).

[4]cf. section A.9

#### Ex Ante Expectation

Before starting with the evaluation, subjects have been handed out a brief description of what HypGen is about and its core capabilities (ca. 130 words). Based on this, they were asked when they expect to use such a tool as part of their product management work and why (Q4, open question).

The mentioned scenarios to 'hire HypGen'[5] as a Product Manager can be condensed as:

- prioritizing where to focus next in product development
- deriving product-related hypotheses from data
- using it for any iteration to improve the product (feature change, information architecture, ...)
- deciding between multiple alternatives to solve the same problem
- organizing experiments and documentation for future use
- communicating results to leadership (which product changes worked and which did not)

To conclude, subjects expect to hire HypGen for the scenarios it was built for, especially the use for each relevant product change or iteration. Noteworthy is drawing on HypGen's information to communicate with leadership about winning tests, failed hypotheses, etc. which places the expectation beyond a workflow tool that is restricted to Product Managers only.

### 7.2.2. Overall Research Question

After the prototype demonstration, subjects were asked two core questions to validate if HypGen can have a profound impact on helping them to improve in hypotheses generation (cf. Figure 7.2).

First, it is tested if subjects think the DSS helps them to define product changes more likely as statistically testable hypotheses (Q5). Subjects responded very positively, where 3/5 subjects strongly agreed to this notion and 2/5 agreed. From qualitative follow-up questions, one can conclude that subjects especially found it helpful that the tool educated them on what to consider for hypotheses generation and guided them step-by-step from target metrics that should be improved to the experiment to ultimately test the hypotheses.

Second, beyond formally sound hypotheses, subjects have been asked if they believe HypGen helps them in defining more valuable product hypotheses (Q6). Again, subjects responded very positively, where 2/5 strongly agreed and 3/5 agreed. A wide variety of features was mentioned in the follow-up question that seems to positively influence

[5] Hiring a tool is a metaphor from Christensen's job to be done framework [143] which describes that solutions are hired to get a job done. In this case, the question aims at which jobs do subjects think they get done with HypGen.

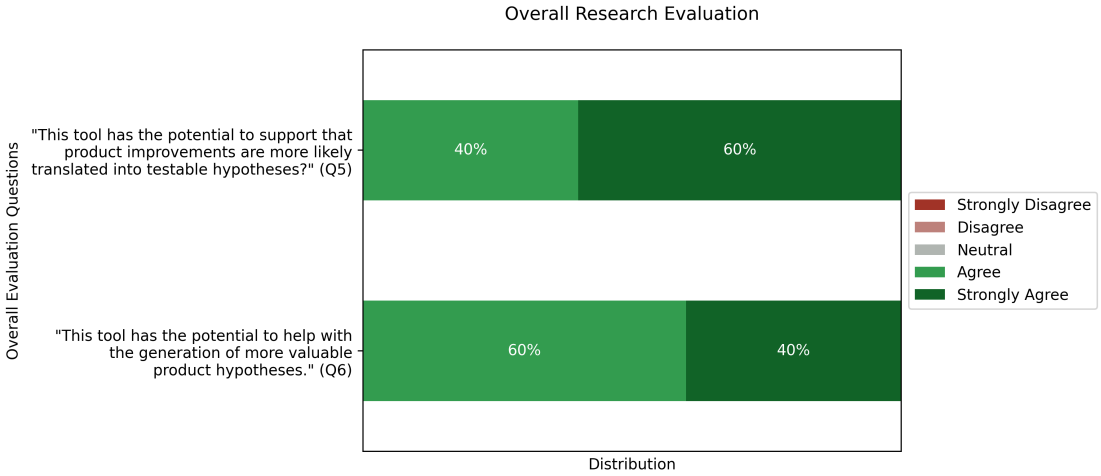



Figure 7.2.: Overall Research Evaluation

the value of hypotheses. Thinking more broadly about each product area (happiness, engagement, retention, adoption, task success) seems to give more clarity about where to likely improve the product most effectively. Having suggestions on what to change in the product inspires critical thinking and provides an outsider perspective to the Product Manager which was mentioned several times. Further, the step-by-step workflow forces the Product Manager to think more deeply about the hypothesis itself.

To summarize the implication of both dimensions, HypGen seems to help in both 1) formally describing feature changes as statistically testable hypotheses and 2) generating more valuable hypotheses.

### 7.2.3. Artifact's Evaluation Criteria

#### Accuracy of Predictions and Analysis

To check the performance of HypGen with respect to the accuracy of its metrics calculations and sample size predictions, unit tests have been conducted. Sample event or survey data sets have been provided and HypGen's algorithms have been passing all unit tests. As sample size estimations are not trivial, different external websites to estimate sample sizes for experiments have been taken as benchmarks. Testing multiple combinations of values for MDE, statistical power, and statistical significance, HypGen's duration and sample size estimations have been tested again with a set of unit tests. All tests have passed, i.e. they did not differ more than 15% in any given configuration from the estimations in public sources [144, 145].

#### Applicability Beyond Case Study

To understand further if HypGen has the potential to provide value beyond the case study's application, subjects have been asked how likely they are to recommend HypGen

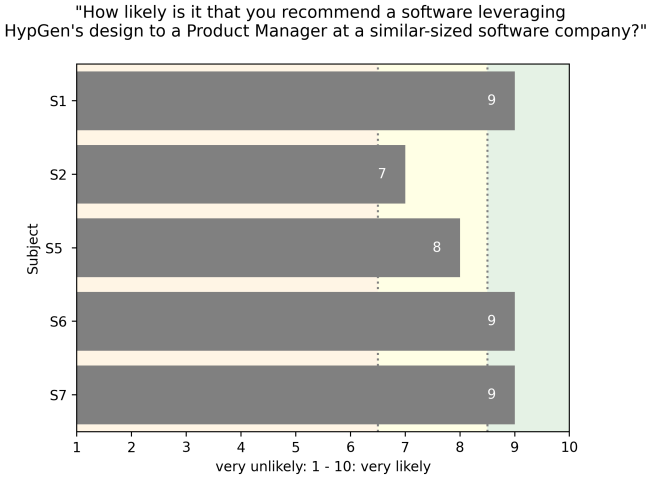



Figure 7.3.: HypeGen Recommendation

to other Product Managers at other companies in similar environments (Q7, cf. Figure 7.3). The question is framed in a way to allow for Net Promoter Score (NPS) calculation [146]. All subjects answered with at least 7/10, which indicates there was no negative sentiment with regard to HypGen. In NPS terms, 2/5 subjects would be classified as 'passives', and 3/5 would be classified as 'promoters', resulting in an overall positive NPS score of +60[6].

When asked further what motivated subjects towards their answer, they responded that HypGen can help Product Managers in every maturity level, also without much prior knowledge. Especially for smaller companies with inexperienced Product Managers, this can help to make their work more hypothesis-driven and improve their professionalism. Improvement potential in the UX was one key aspect for a subject to give an answer which classifies him as 'passive'.

### Additional Evaluation Criteria

To understand further dimensions of the DSR artifact, the researcher has selected seven further evaluation criteria (cf. Figure 7.4) and asked subjects to rate to what degree they agree to a statement about each fitness criterion[7]. Optionally, subjects were able to provide further qualitative information per criterion. Results per criterion are briefly presented.

**Business Alignment** gauges how far HypGen can contribute to greater adoption of CEx. 4 out of 5 subjects agree or strongly agree that HypGen can contribute to greater adoption, mostly because of its in-app education and making creating experiments easier

[6]Calculation of NPS score: % promoters - % detractors (detractor: 1-6; passive: 7-8; promoter: 9-10; in this case 3 subjects are considered promoters (60%) and none as detractors which results in a NPS score of +60)

[7]The exact statements are documented in the Appendix (cf. section A.9)

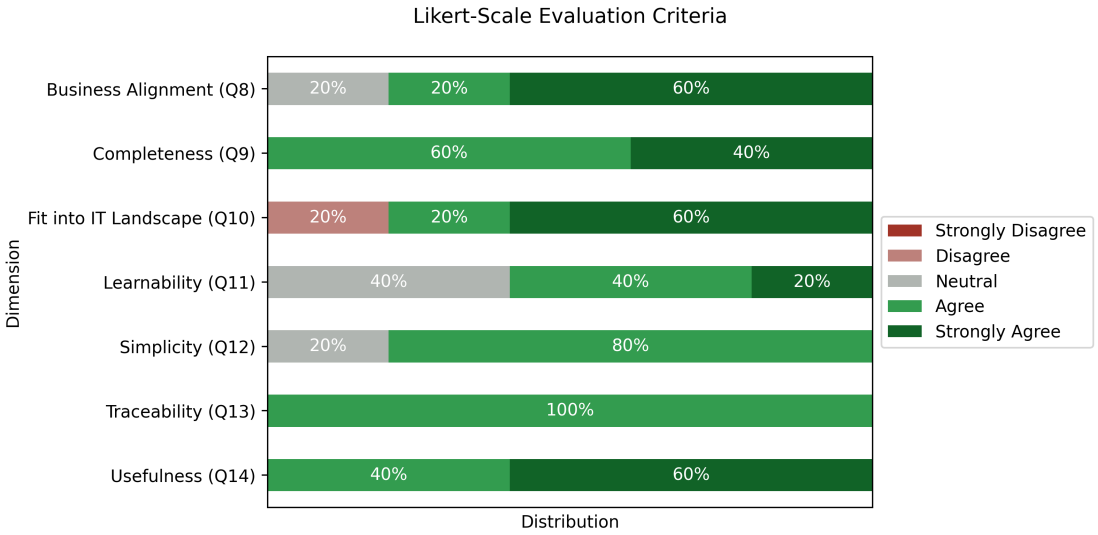



Figure 7.4.: Artifact's Evaluation Criteria

by generating the needed hypotheses. The neutral opinion questioned how far following the guidance meticulously brings value.

**Completeness** aims to find out how exhaustive HypGen is perceived to include the most important aspects to define testable and valuable product hypotheses. Subjects consistently report they find it as complete as it seems to cover most steps that are needed in the course of hypotheses generation.

**Fit into IT Landscape** measures how well HypGen fits into the tooling landscape that Product Managers are using today. While 4 out of 5 subjects perceive it as fitting into the IT landscape and mention the good look and feel and professionalism, it is pointed out that focusing on integrations for better fit into the IT landscape is very important for such a tool. Moving to a live integration with product analytics tooling as well as tools for incoming customer feedback might be valuable options to improve this dimension.

**Learnability** describes the perception of subjects how easy it is to learn about HypGen's features and how it supports hypotheses generation. Feedback shows overall positive (3 agree or strongly agree), but also mixed perception (2 are neutral). While the clear workflow and tooltips help, descriptions for in-app guidance are sometimes quite long and complex to understand.

**Simplicity** describes a slightly different category than Learnability, i.e. the ease of use of HypGen. Subjects rate HypGen as easy to follow (4/5 agree to its simplicity) with its guided workflow and context help. Some parts seem to be more straightforward than others, with increasing complexity towards the end of the workflow (e.g. documenting the hypothetical experiment's outcome was seen as a challenge). Improvement areas in the user experience have been mentioned which in turn have also been described as easily fixable.

**Traceability** aims to understand how far HypGen contributes to unambiguously fol-

lowing the path from business opportunity to released software functionality. 5/5 subjects were positive about this category with no further mentions besides that documentation is always helpful in the work of product management.

**Usefulness** measures to what degree HypGen really helps in defining effective, testable product improvement hypotheses. While all subjects perceive it as useful, 3/5 subjects perceive HypGen as very useful, i.e. strongly agree. Subjects mention that they believe HypGen fosters scientific rigor in product management and eases hypotheses generation. A subject stated that once hypotheses are generated, this also enables the possibility of more experimentation in the future. To cite a subject, HypGen 'levels you up to become a Senior Product Manager' with regards to the data-driven dimension (S6).

In summary, subjects rate all performance dimensions overwhelmingly positive. For some dimensions (e.g. *Usefulness*, *Completeness*, *Traceability*), subjects tend to agree consistently. Especially *Learnability* and *Fit into IT Landscape* tend to receive the relatively most critical answers which in turn yields the two major improvement areas for HypGen.

## 7.2.4. Investigation of Core Value

### Perceived Value of Capabilities

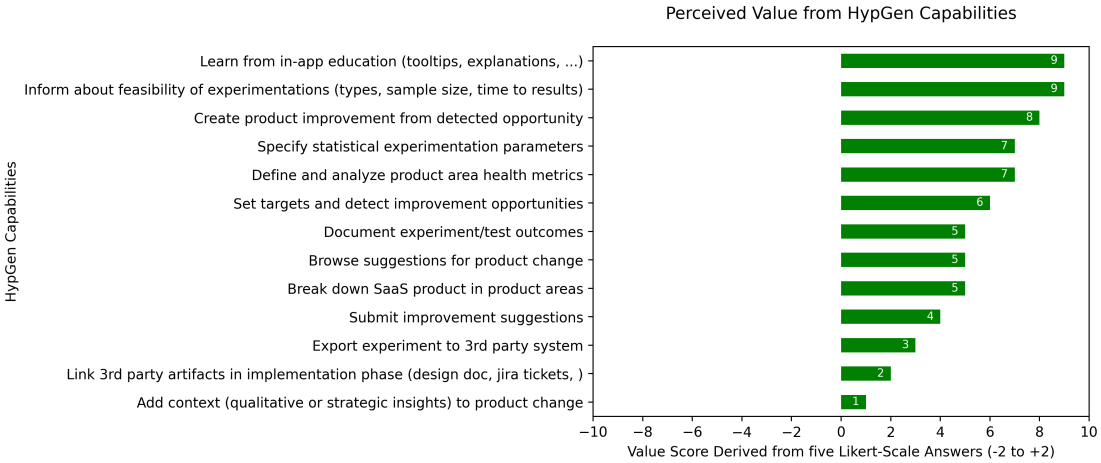



Figure 7.5.: HypeGen Capabilities and Their Perceived Value

Subjects have been asked (Q15): *'How useful do you rate each of these selected HypGen capabilities with respect to adding value to your workflow when defining impactful, testable product improvements?'* on a 5pt Likert-Scale of -2 (irrelevant) to +2 (useful). For simplicity, the author creates a score summing up the results from all subjects and ranks HypGen's capabilities from most valuable to least valuable. The maximum score is +10 (5 subjects giving a +2 (useful) answer) or likewise -10 (5 subjects giving a -2 (irrelevant) answer).

Results (cf. Figure 7.5) show that all listed capabilities are ranked rather useful than irrelevant. This indicates that all provide at least some value in the bigger picture of hypotheses generation. However, it can also be clearly seen that some capabilities might contribute more value to Product Managers. Especially relevant for subjects seems to be HypGen's in-app education as well as context helps to check if hypothesis testing is feasible in a given experimental setting. Further, creating the product improvement directly from a detected opportunity (with preceding health analysis) seems to offer great value. Getting baseline values for a target metric and a backing-up data analysis seems to make life easier for Product Managers to get started with generating the hypothesis.

While still being perceived to offer some value, aligning the product change to strategic considerations and research insights seems to not offer too much value. Second, linking the hypothesis to implementation or exporting the experiment is rated as offering only limited value. As the documentation of the experiment/test outcome is perceived by many as rather useful (which kind of needs a linkage to an experiment or third-party tool), these aspects might speak to making the solution a bit more lightweight towards the implementation and review/learn phase of the experiment.

### Dwell Time in Areas of Interest

Areas of Interest (AoIs) are a way to better group interactions with software systems into logical parts of the user interface [142]. The author has mapped each of the almost 100 interaction event types (any relevant clicks on UI elements, keystrokes in input fields, tooltip hovers, ...) to one out of around 20 AoIs[8]. Doing so, one can find out where subjects spend the most time interacting with the software on a higher level, which gives a good indication of the software's core (cf. Figure 7.6).

From analysis, it can be clearly seen that subjects spend the most time in the hypotheses generation relevant AoIs, i.e. to define the hypothesis (*PI - Hypothesis*) and select appropriate experimentation (*PI - Experimentation*). Also specifying different analysis types consumed quite some time from evaluation subjects (can be identified in the prefix *Configure Health* in multiple AoIs). Reviewing improvement opportunities (two AoIs: *Improvement Opportunities*, *Improvement Opportunities - Analysis*) did not consume that much time from subjects. This seems plausible as it involves tasks about reviewing and selecting based on ranked suggestions. Subjects spend much less time documenting outcomes or linking implementation artifacts than generating the hypothesis itself and deciding on a feasible experimentation setup. This gives further backing for the notion that HypGen nudges Product Managers to take significant time to think about hypotheses including their experimentation before already jumping to feature implementation.

[8] The list of AoIs as well as a high-level mapping of AoIs to the UI can be found in the Appendix (cf. section A.10). The concrete mapping between each AoIs and the exact interaction event type (e.g. which button exactly is mapped to an AoI) is part of the code documentation on GitLab.

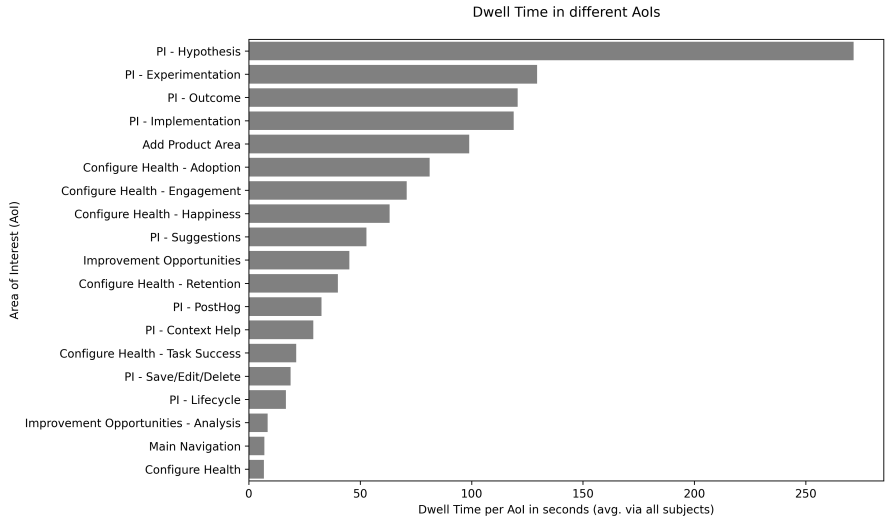



Figure 7.6.: Dwell Time in Different Areas of Interest

### 7.2.5. Task Performance

As illustrated before, subjects had to conduct a sequence of five tasks which all build on each other. Task performance will be analyzed in multiple ways. First, subjects are asked how they rate the *ease of task completion* for each task on a scale of 1 (very difficult) to 7 (very easy) and are also motivated to provide more context on their choice (Q16-Q21). Second, *time to task completion* is measured from reading the task until finishing the task. Following, an overview and comparison between tasks on a higher level, deep dives into each task are conducted.

#### Overview

When analyzing *ease of task completion* (cf. Figure 7.7), insights can be generated both from comparing tasks among each other (e.g. averages marked with an X), as well as diverging perceptions of different individuals for the same task (individual points, percentiles). For the sake of understanding *ease of task completion* better, the fourth task *'Define an Improvement Hypothesis & Decide for a feasible Experimentation Set-Up'* has been broken up into two parts for analysis. 4a is about the part of defining the hypothesis, while the second part includes the decision for a feasible experimentation setup (4b).

In general, it can be observed that task 1 to create a product area is perceived to be very straightforward to complete (avg: 6.8/7), with 4/5 subjects rating it 7/7. On the other side of the spectrum, subjects found it comparatively complicated to solve task 5 (avg: 4.4/7). When analyzing qualitative follow-up responses, it can be traced back largely to the hypothetical nature of the task. Subjects needed to make up a theoretical experiment outcome in the future while the experiment did not start in real life yet. As

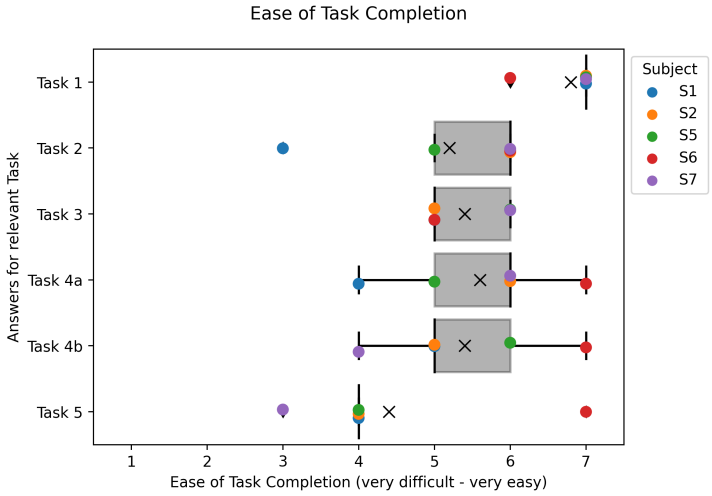



Figure 7.7.: Ease of Task Completion

moderately difficult, task 2[9] (avg. 5.2/7), task 3[10] (avg: 5.4/7), task 4a (avg: 5.6/7), and task 4b (avg: 5.4/7) can be identified. It can be seen that they were roughly comparable in task complexity while each task has yielded some challenges for subjects. Noticeable are also larger outliers for task 2, tasks 4a, and 4b, as well as task 5 which will be discussed later on. When checking the overall perception across all tasks among individuals, one can observe that both the mean and median answers of each subject are above 4 (the neutral answer). This indicates that all subjects have perceived the tasks as rather easy than hard to complete.

Alongside the ease of task completion, *time to task completion* can contribute a similar, yet different angle (cf. Figure 7.8). The author measures the *time to task completion* from the moment subjects start reading the task prompt until the moment they have completed the task[11].

It can be clearly observed that the tasks each have different lengths to complete. Task 1 (median: 126s) and task 3 (median: 67s) have been achieved quite fast which comes as no greater surprise. Task 1 was described as very easy to complete and task 3 involves reviewing a suggested list of improvement opportunities and selecting a promising one. Task 2 already takes comparatively longer (median: 291s) as it involves critical thinking about how to analyze the product area and which metrics are important. Task 4, although described as straightforward to complete, takes significant time (median: 515s) as again it includes a lot of critical thinking and creativity. In Task 4, subjects need to come up with a strong hypothesis while reviewing product change suggestions and grounding the hypothesis in research and strategy. In addition, subjects need to configure statistical

[9] Specify product health metrics
[10] Review and select improvement opportunities
[11] Task 2 said to configure 2 or more HEART dimensions to measure product health, thus time to complete the first two dimensions has been drawn upon for fair comparisons

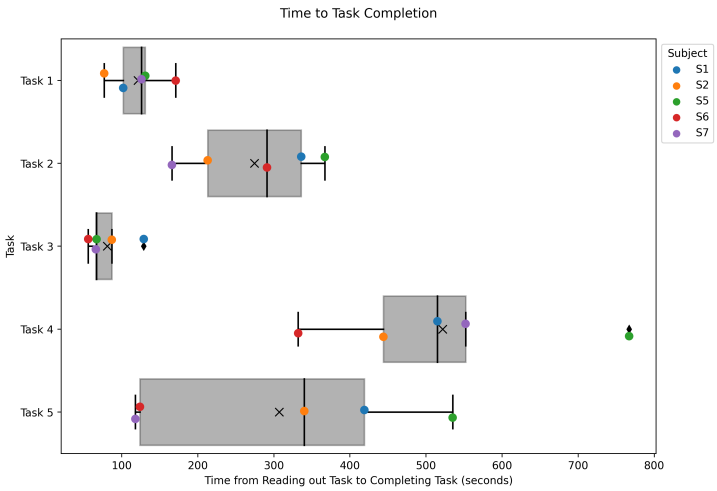



Figure 7.8.: Time to Task Completion

parameters and choose a corresponding experiment. Task 5 also takes a while (median: 340s) to complete with great deviations among subjects. Some took the hypothetical scenario of coming up with an experiment outcome and documenting it way more seriously than others who have made quick assumptions. Differences in overall task completion time do not only hold true for the last task. When observing the total time to complete all tasks, it can be seen that some subjects have completed all tasks in less than 20 minutes (S2, S6, S7), whereas others spend significantly more time (S1, S5).

When analyzing *time to task completion* and contrasting it with *ease of task completion*, one can derive interesting insights. First, *time to task completion* is not within the low seconds area which comes as no surprise as the tasks are quite complex in nature and require contextual thinking of Product Managers as well as learning about hypotheses generation experimentation on the go. Second, *ease of task completion* and *time to task completion* are two complementary dimensions when analyzing HypGen. HypGen is not a tool that is meant to just speed up the process of hypotheses generation. Its main benefit lies in guiding the user to think along a process with scientific rigor, hence motivating them to apply critical thinking to avoid jumping to conclusions too early. This requires time to be spent in the tool, thinking step-by-step, and leveraging HypGen's context helps, suggestions, and in-app education. This duration can be different for each Product Manager and might also depend on the product area and opportunity of choice (some hypotheses might be more twisted, others might be more straightforward).

### Task 1

To analyze individual task-solving behavior, UI click event analysis, mouse location analysis, as well as qualitative information (qualitative survey answers, think aloud) to solve the tasks is leveraged. The author is drawing from the available options to analyze

each task best while omitting analyses that do not add much value[12].

In task 1, subjects needed to start from an empty state to *'create a product area incl. tracking events'*. All subjects realized they needed to click on 'Create Product Area' to navigate to the right page (*navigate*) and then started to fill out the two needed fields (name for the product area) and select a number of tracking events (cf. Figure 7.9).

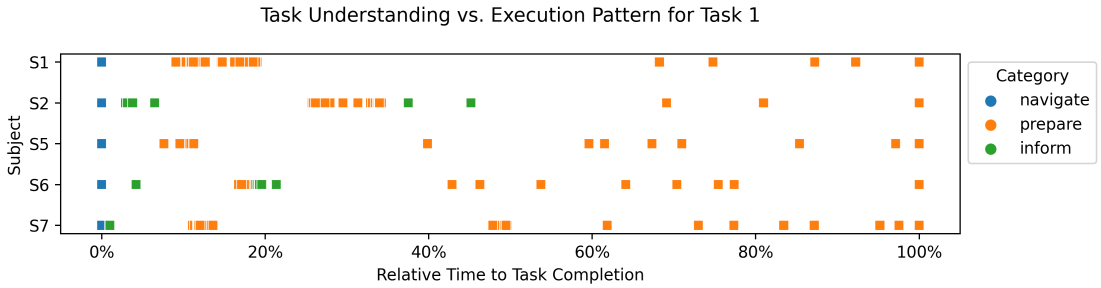



Figure 7.9.: Interaction Plot: Task 1 Exploration vs. Exploitation

Subjects quickly informed what HypGen understands as tracking events (*inform*), explored the list, and got the job done (*prepare*). However, still, two interesting facts can be observed. First, all subjects have been defining realistic real-world product areas of their company with each a selection of multiple events to capture user behavior. This prepares them to evaluate HypGen in a very realistic setting. Second, while all subjects had an intuitive understanding of what a product area is and what it means for their business, two potential pitfalls appeared. One subject started to define a product area for which no events are tracked yet. The subject was surprised, however, for this part of the software there was no event tracking set-up. This made it transparent that product telemetry with event tracking is the foundation to work data-driven and use HypGen effectively. The subject then continued to select another product area where events are tracked extensively. On the other side, subjects occasionally struggled to think of a good product area before having a look at all the captured events to choose from. This may lead to the assumption that flipping the logic to first selecting a subset of events and then giving them 'a headline' (the product area) might lead to an even easier experience.

### Task 2

In task 2 (cf. Figure 7.10), subjects have been asked to *'specify product health metrics'* for their recently created product area in task 1.

Having a look at interaction plots, one can observe interactions with different AoIs in the course of completing task 2. On an aggregate level, one can see that subjects usually first start with the 'Configure Health' AoI, which is functionality that gives users an idea of what HypGen defines as product health and how it relates to its five sub-categories

[12]E.g. the author omits interaction plots once the tasks only involve 1-2 AoI, or omits mouse location analyses once they do not answer an important question for understanding a task better

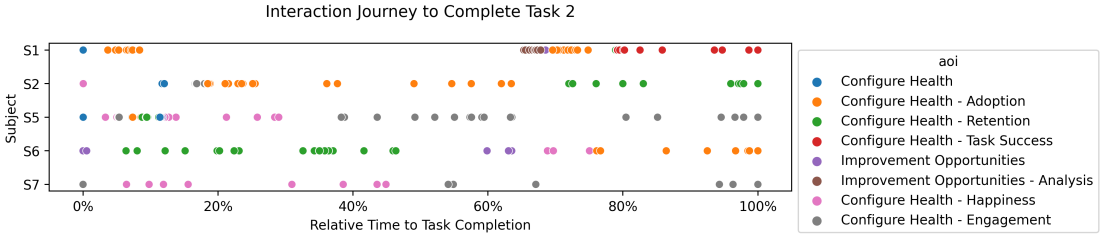



Figure 7.10.: Interaction Plot: Task 2

(H-E-A-R-T). Subjects then purposefully decide on health dimensions which are key for their product area. When zooming out of individual interaction graphs, one can see that all five dimensions of HEART have been selected at least once by any subject, as can be seen in the legend. Adoption has been selected most often, i.e. by 3/5 subjects[13]. Some subjects explored to define a health dimension but then continued to another one as they deemed it to fit more. One example included a Product Manager (cf. S6) who attempted to analyze happiness but found out that there was no survey available that asked real users to vote on satisfaction with the given product area. This again made it transparent that product telemetry and primary data such as surveys are the starting point and play a significant role in how much value to extract when working with HypGen. After subjects have defined their first health dimension with the corresponding metric, they are presented with the improvement opportunity for this dimension. Some subjects immediately interacted with the analysis to get more details (cf. S1) while others continued straightforward to the second health dimension.

### Task 3

As task 3 includes in its core reviewing a sorted list of improvement opportunities and optionally reviewing the analysis, the author reports mouse heatmaps (cf. Figure 7.11) accompanied with qualitative observations.

Once reviewing and deciding on an improvement opportunity, subjects have been reviewing the suggested and ranked improvement opportunities as part of data analysis in the background. All subjects selected highly ranked improvement opportunities - but they also informed themselves about what was analyzed, including how many samples were part of the analysis. Using their domain knowledge, Product Managers did not always select the top-ranked opportunity but e.g. chose the second-ranked (e.g. S1 who chose opportunity ranked 2 out of 8[14]).

The usual flow can be illustrated quite nicely with the recorded mouse locations[15] of

[13] In the interaction plot, the author observes subjects until completing configuration of two health dimensions

[14] This was a subject who voluntarily configured all HEART categories, hence receiving more opportunities suggested

[15] The User Interface in the background shows an anonymized version of the exact case the subject used (also had 2 improvement opportunities)

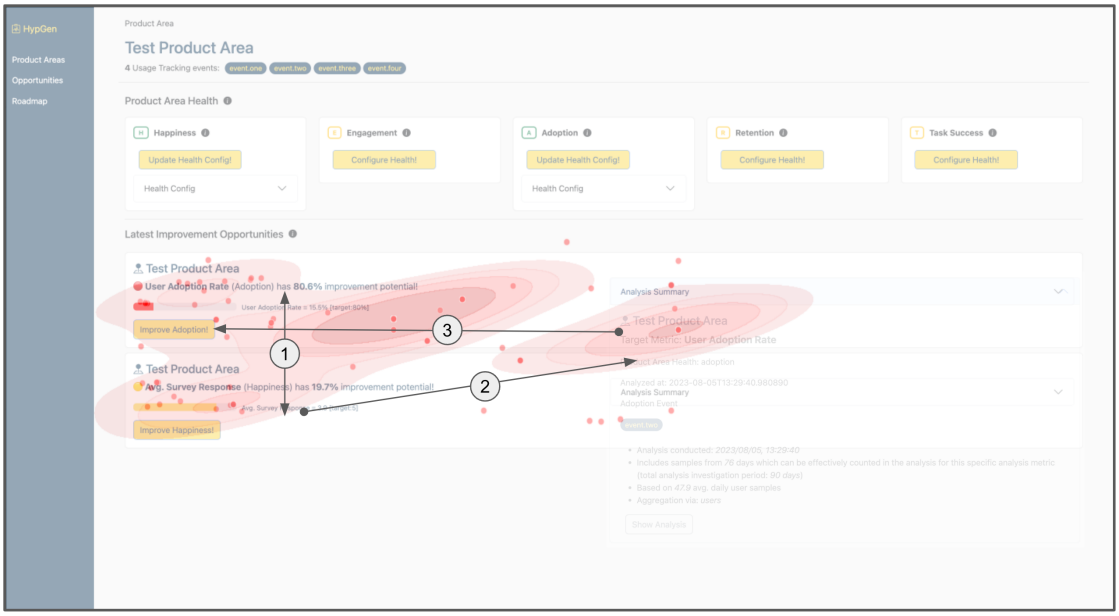



Figure 7.11.: Mouse Heatmap: Task 3 (Subject S5)

subject S5. The subject was comparatively reviewing the improvement opportunities first (marked with 1, scrolling up and down). Then, she realized it made more sense to optimize the higher-ranked opportunity, but was curious to find out more about the analysis (2). After reviewing the analysis results, she selected the top-ranked opportunity to continue improving it with a hypothesis (3).

### Task 4

To recall task 4, it was prompted to *'define an improvement hypothesis & decide for a feasible experimentation set-up'*. When looking at the interaction journey, interesting patterns appear (cf. Figure 7.12). First, users start to define the hypothesis' product change. They then experience they can be supported by HypGen's suggestions (*PI - Suggestions*). After browsing suggestions for a while, one can see that they then jumped back to the hypothesis AoI to fine-tune the hypothesis. 4/5 subjects discovered and all of them have actually used a suggested product change. Whereas 2/4 subjects took over the whole suggested product change, the other 2 included large aspects of the suggested product change but tailored it to their specific context. In doing so, they blended the suggestion with their domain experience and creativity. They proceed with filling out multiple supporting fields to make the product change more robust and select the MDE. Then, subjects switched to selecting appropriate experimentation after around 40%-65% of the task completion time (*PI - Experimentation*). Again 4/5 subjects leveraged the experimentation context help (*PI - Context Help*) which educates them about the pros and cons of each experimentation set-up and delivers them sample size and time to

results estimations. After learning about these facts and often experiencing too much time until receiving experiment results, some subjects went back to the hypothesis to change it slightly (cf. S2, S5) to decrease the time the experiment needs to run - while e.g. pushing for a higher MDE (a hypothesis with a greater leap of faith) or accepting less scientific rigor (e.g. decreasing statistical significance). After that, subjects go back to the experiment and finish its planning.

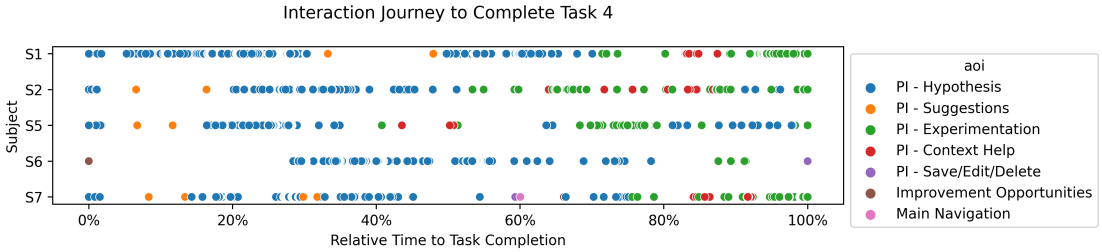



Figure 7.12.: Interaction Plot: Task 4

To shed light on learning behavior in hypotheses generation's core activities, the author groups interactions with HypGen into three meta categories: navigate, inform, and prepare (cf. Figure 7.13). Navigate refers to just interactions with the purpose of clicking through a modal or using the main navigation. Inform refers to parts where users are using education tooltips, context helps, or review analyses or suggestions by HypGen. Prepare refers to filling in information or to selecting values from dropdowns. One can clearly observe the iterative behavior of informing about each step and subjects learning on the go. Subjects seem to explore and learn and then immediately apply this knowledge to conduct a subsequent action.

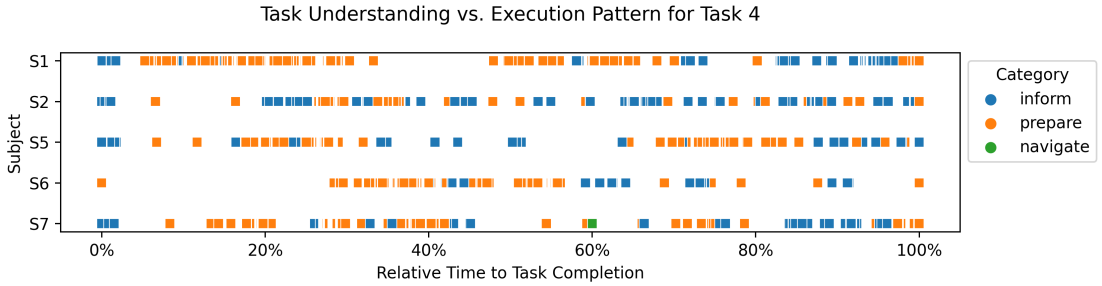



Figure 7.13.: Interaction Plot: Task 4 Exploration vs. Exploitation

### Task 5

In task 5, subjects needed to *'simulate implementation and further steps until the hypothesis outcome'* based on the previously selected hypothesis and experiment combination (cf. Figure 7.14).

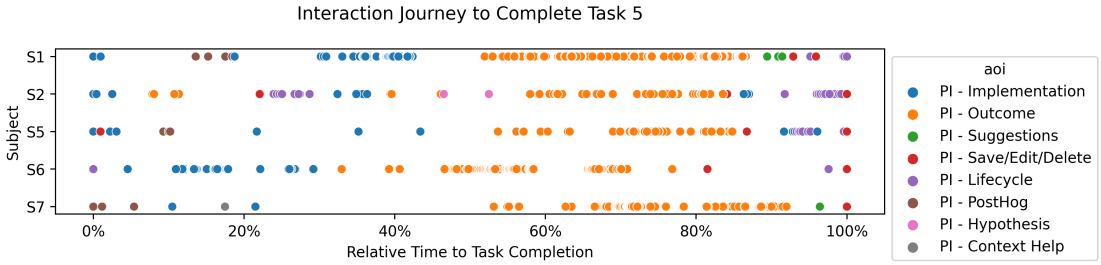



Figure 7.14.: Interaction Plot: Task 5

First, to simulate implementation, one could link external documents (e.g. Analytics reports, GitHub PRs, or further feature documentation) as well as export the experiment (if it was a randomized controlled experiment - A/B test) to the experimentation tool PostHog. Some subjects that opted before for a randomized controlled experiment have both discovered the PostHog integration and also synced the planned A/B test from HypGen to PostHog. Others have continued with linking other tools to HypGen (without an integration). The tools they thought of linking HypGen with were mostly Tableau for Analytics and Confluence for further documentation of the product change to develop[16].

Second, to document a theoretical experiment outcome, subjects spend some time filling out the necessary information while informing in each step what is required. To think about the outcome, some subjects (cf. S2, S5) went back to the original hypothesis which seems they would have admired to see the hypotheses next to documenting its outcome. After defining everything for the outcome, not all subjects realized they had the option to save the outcome as a future suggestion (suggesting a product change for the next time working on a similar improvement opportunity). However, others (e.g. S1) explored this special feature. Once having completed information input, most of the users saved the product improvement and then realized that it had not yet transitioned into the final stage of its lifecycle (i.e. Learn). To resolve this, they attempted two strategies. One user tried to drag and drop the hypothesis card to the Learn phase like he is used to from other task board tools. However, this functionality was not implemented and he stopped there. Others pursued another strategy which was clicking again on the detail page of the product improvement, browsing around until finding the 'lifecycle switcher'. Apparently, the lifecycle switcher was not easy to detect so subjects spent some time finding the right element and finally transitioning the product improvement, i.e. the extended hypothesis to the Learn phase.

## 7.3. Summary

In the previous pages, the author has conducted the step-by-step evaluation of the overarching research question, artifact evaluation criteria, and the extent to which they

[16]These two tools have already been collected as core tools in the Product Manager IT landscape as part of the interview study

were met by HypGen. Additionally, the author has conducted an in-depth analysis of task completion behavior as well as an exploration of where the core value of HypGen is located. In this section, the author summarizes the results while implications are discussed in the closing chapter of the thesis.

Evaluation subjects report before HypGen's demonstration that they do not define product improvements as testable hypotheses often, nor do they engage frequently in experimentation. After engaging with HypGen in a realistic setting (analyzing their own SaaS product with real usage and survey data), subjects report that HypGen can improve the likelihood that they will define features as testable product hypotheses in the future. Further, they report that HypGen can help them to generate product hypotheses that are likely more valuable and impactful. This yields indications that HypGen can support beyond achieving a formal definition of a hypothesis, but by improving the value of hypotheses that are ultimately tested as part of experimentation. This demonstrates that HypGen does not just support the Product Manager to translate any intuitive idea into a statistically testable hypothesis. In addition, HypGen fosters critical thinking with many (data-driven) capabilities offered. Further, the study finds that a hypotheses generation DSS would likely not be just used for a few moments to brainstorm a hypothesis, but the DSS would create a setting where Product Managers engage thoroughly with a range of functionality to optimize a hypothesis for feasibility and large impact.

When it comes to applicability beyond the case study, HypGen receives an overall positive NPS score of +60 (n=5). Results indicate that such a design can potentially help SaaS Product Managers in other, yet similar settings as well. Moreover, subjects have reported that a DSS such as HypGen can empower a greater adoption of experimentation which feeds the impression that thought-through design of a hypotheses generation DSS which builds the bridge to experimentation can have an overall positive effect towards adoption of experimentation in the industry. In order to create a link back to the interview study, it was identified that Product Managers did neither really work with product hypotheses nor do they engage in rigorous experimentation. With evaluating HypGen in the environment, the problems identified in the pre-study with respect to hypotheses generation and experimentation can be understood to be successfully addressed, at least in the specific setting of the case study.

While HypGen is perceived as solving to some extent the problem of 'most companies [...] rather rely solely on gut feel and intuition to make product decisions' [56], the evaluation yields far more insights that add to the body of knowledge. From this DSR case study it becomes apparent that a step-by-step workflow to generate hypotheses accompanied with suggestions, contextual views, and informative tooltips as core design philosophy is perceived well by Product Managers. The research has proven that a DSS can guide Product Managers at the case study company in hypotheses generation's decision-making steps in an end-to-end fashion. This starts with which problems to best fix in a product area and continues to generating the hypothesis, choosing feasible experimentation, to finally monitoring the decision's outcome. In support of this, especially 'connecting the dots' functionality received high-value points when interviewees were asked which of HypGen's capabilities they found most useful. Perceived as highly useful

have been the predictive connection of hypothesis and experimentation where for a given hypothesis an estimation for sample sizes and experiment duration was given. Further, connecting the dots between a low-performing product health metric to creating the product improvement hypothesis with a single click and already benefiting from past knowledge (standard deviations, daily samples, baselines) has been perceived as highly useful.

# 8. Conclusion

In DSR projects, contributions of a project can and should be manifold. On the one hand, the artifact itself contributes a solution to a previously unsolved problem in the environment of study. On the other hand, all efforts in exploration and evaluation contribute insights to the body of knowledge. This closing chapter is structured as follows. First, different kinds of contributions to the knowledge base are discussed. Continuing, the whole thesis project is reviewed in the frame of high quality DSR. Finally, the paper concludes with limitations to the research as well as possibilities for future research.

## 8.1. Contributions

### 8.1.1. Relevance in Application Domain

When it comes to design science's relevance cycle, a DSR project needs to show its relevance by creating an artifact that solves a problem in a chosen environment successfully [58]. For the B2B SaaS company and its intertwined tasks of feature decision-making, it could be demonstrated that the developed artifact HypGen can effectively support the generation of statistically testable product hypotheses that are aimed at creating valuable impact. The evaluation showed that interviewees both perceive HypGen as a) empowering more feature decisions to be translated into statistically testable hypotheses, and b) that those hypotheses are perceived to be more valuable with HypGen's help than without. The artifact's support to generate hypotheses for software product's most pressing improvement opportunities as well as suggesting potential product changes and many more capabilities have been deemed helpful for deciding on more valuable hypotheses.

Beyond proving relevance for the overall research question, the research reveals further insight. By empowering hypotheses generation and its connection by design to experimentation planning, HypGen is perceived to have a positive effect on CEx adoption. This insight creates optimism for other companies in the B2B SaaS ecosystem that might not be very mature in experimentation or hypotheses generation yet. In fact, research subjects in this project tend to recommend HypGen to other Product Managers working in similar environments.

### 8.1.2. Addition to Knowledge Base

Having presented the contributions that can be described in the relevance cycle, this section focuses on the contributions with respect to the rigor cycle [58]. A project that creates an artifact to solve a problem in the environment without improving innovation in

a field does not satisfy the objective of design science. This delineates DSR from applying routine design [1] where contributions to design knowledge are not in focus.

### Descriptive Omega Knowledge

Contributions in descriptive knowledge in the context of DSR projects describe those contributions that shed light on general principles and patterns that seem to be present in the environment under study [2]. The explorative interview study generates descriptive knowledge about hypotheses generation as presented in chapter 4 in greater detail. Below, the author identifies three core descriptive knowledge contributions to the research landscape which bring more nuance to some topics.

Firstly, an important contribution is likely the portrayal of intuition, creativity, and experience not solely as a flaw in the system of generating product hypotheses. But, this study finds value in intuition, creativity, and experience as a core necessity in the presence of infinite search spaces. Research often frames intuition as highly opinionated and sheds a negative light on it (cf. Kohavi's HiPPo narrative [15]). The criticism of intuition in the system certainly has justification in those parts where intuition can be rationalized. However, it is unlikely to assume reasonable decisions on which hypotheses to test in experiments with solely relying on automation and excluding the human. Too many tacit and informal aspects of hypotheses generation are encountered in the decision-making process.

Secondly, by providing insight into the complexities of product change decisions at smaller B2B companies, the study adds more nuanced insights to the industry-leader-dominated knowledge base. Defining 'if a, then b' statements might look like what hypotheses generation is about from a superficial perspective. However, this is perhaps more appropriate in highly mature products where the core functionality is already stable and small change experiments (button colors or call-to-actions) are conducted. The interview study shows that B2B Product Managers in startups tend to have more twisted product problems to solve that require basing their decision-making on condensed insights from qualitative and quantitative research as well as blending it with their own company's strategic considerations. This also results in shipping larger changes that aim to improve often low-performing product areas. Taken together, this portrays a complex setting where hypotheses generation is not straightforward, but where on the other hand product changes' effect sizes might be larger.

Thirdly, the study contributes another aspect to the scientific debate by suggesting a core of the CEx adoption problem in the perfectible formalization of hypotheses and its often missing link to experimentation. Reviewing literature on *Hypotheses Engineering* already shows rather vague hypothesizing practices. In support of that, the interview study finds that subjects interchangeably use the words assumptions and hypotheses by describing something that has not yet been tested. However, their assumptions or hypotheses are not described in a fashion that allows for statistical validation. So in order to quasi-validate their assumptions, Product Managers often search exploratively for clues after release if an assumption has been true or not. Unsurprisingly, this creates the risk of confirmation bias which ironically is a bias that Product Managers are well aware of

and try to mitigate.

In summary, the interview study motivates an extended perspective on the hypothesis in CEx. Besides the product change (if a) and its desired effect on a target metric (then y), considerations of why the product change will drive the target metric and why this makes sense for the company strategically may need to be taken into account. Beyond that, also early risk mitigation should be part of a broader hypothesis definition (e.g. user testings) as well as higher-level experimentation planning. Because less mature products also have higher potentials for improvement, a key focus of hypotheses generation at a startup also revolves around detecting gaps in product health and aiming for larger effect sizes in experiments than normally encountered at industry leaders.

### Prescriptive Lambda Knowledge

The knowledge generated via the development of artifacts and their evaluation can be classified as lambda, or prescriptive knowledge [2]. On a high level, the development and evaluation of HypGen could successfully answer the research question: *'How should a Decision Support System be designed to improve the generation of testable and valuable product improvement hypotheses in a SaaS startup context'*? All subjects in the evaluation successfully generated hypotheses with HypGen and they perceived those hypotheses also as valuable, i.e. they are not just any idea translated into a statistical hypothesis. To go into more detail, when observing task success while generating hypotheses with HypGen and blending it with Product Managers' perceptions, valuable design knowledge can be generated. To avoid overwhelming the reader with all possible design insights, the author gives a non-exhaustive list of eight important design knowledge contributions. The interested reader may mentally add her inspirations for knowledge generated to this list.

First, building a software that combines decomposing a software product into product areas and analyzing them with Google's HEART framework as starting point for hypotheses generation can be seen as an effective design. Schermann et al. conclude in their work that 'practitioners are in need of more principled approaches to release decision making (e.g., which features to conduct experiments on, or which metrics to evaluate)' [20]. This research could prove that HypGen successfully detects which features (or product areas) to best conduct experiments on and gives guidance regarding which health dimensions and metrics to best take as a target metric. Exceeding this, even more nuanced insights for future design are generated. As of now, one highly configurable metric per HEART dimension has been provided with opportunities for way more metrics to come. However, the corresponding complexity needs to be factored in for future design. Especially when a seemingly easy metric health category (e.g. retention) creates multiple potential target metrics (e.g. short-term, mid-term, long-term retention), interviewees had a complex experience reviewing metric improvement potentials of the same parent category[1]. To summarize, building on a metric framework like HEART and attaching metrics to each category may be a good path to follow, but further resolving target metric complexity by simplifying design is key.

[1]Same goes for analyzing Conversion Rate in a 4-step funnel, which leads to 3 different conversion rates

Second, spotting improvement opportunities based on gap analysis seems to be a reasonable foundation for suggesting and ranking opportunities. Doing so, HypGen quasi-validates core aspects of the HYPEX model where the authors Olsson and Bosch propose a process model for CEx which builds in its core on gap analysis between expected and real product performance [27]. The current prototype works by cooperating with the Product Manager to select expected target values for metrics to then suggest improvement opportunities with the help of dedicated data pipelines and target-actual analysis. When using HypGen, Product Managers did not always choose the no. 1 suggested improvement opportunity, but another top-ranked one by factoring in their domain knowledge and critical thinking skills. This means that the selection of improvement opportunities is likely not straightforward (at least with this approach) to be automated without the Product Manager in the loop. However, a variety of machine learning models can certainly show even more complex opportunities. But, it must be noted that proposed opportunities also need to be relevant for product improvements. Reporting any anomalies in product usage behavior might just create unnecessary noise. A possible solution going beyond gap analysis will probably also need to include some mechanism to get a search or benchmarking context from the Product Manager.

Third, browsing suggestions for product changes already demonstrated remarkable usefulness. In decision-making's choice phase one of multiple solution approaches is selected [79, 80] . HypGen achieves support of this crucial decision-making phase with an unordered list of proven hypotheses from leaked A/B tests, best practices, or LLM generated ideas that are meant to drive a target metric in the same HEART dimension that the hypothesis addresses. With only one exception, all subjects have been building their hypothesis based on a product change suggestion, hence proving its usefulness. These suggestions can likely be even further improved in future designs by leveraging more context, especially with the help of recent advancements in LLMs.

Fourth, adding context to the product change beyond the mere idea of what to change was perceived as only slightly useful which was surprising. Hypotheses frameworks (cf. [91]) promote adding research insights to a hypothesis to back up its cause-effect relationship. Also, the interview study identified that there are useful factors to think about during hypotheses generation (e.g. research insights, strategic fit). However, evaluation has shown that just encouraging Product Managers to think about these categories and to document their research insights and strategic fit in text fields is probably not enough. Product Managers would like to integrate qualitative artifacts directly into HypGen so that these aspects of hypotheses generation are sufficiently supported. Here, integrating a list of *jobs to be done* which could be addressed by a product change, or automatically searching for *qualitative insights* that are loaded from another system may likely be a valuable design choice.

Fifth, when looking at evaluation criteria, the category of *IT Landscape FIT* created interesting design insights. Integrating with third-party tools to insource data is generally seen as important for DSS [4]. A standalone tool with data being read from .csv files[2] which is not deeply integrated into the product analytics tool itself might lead to adoption

[2]due to the scope and security need as part of the thesis project

problems. At least a live connection (or batch data loading) is very likely a must for working with such a tool effectively in the long run. Even considering rethinking such a DSS to not be a software on its own, but a module or plugin for a product analytics or an experimentation system might be reasonable.

Sixth, it could be seen that all interviewees chose some form of hypothesis testing that is of higher rigor than the post-launch analysis which was mentioned during the interview study. This can prove that HypGen can be considered an effective design to move companies up the stairway to heaven [55], i.e. towards a high maturity in experimentation-driven software development. Although HypGen tries to nudge the user towards applying gold standard testing, 2/5 interviewees still opted for a pre-post test[3] instead of a randomized controlled experiment (3/5) to validate their hypothesis. Future design may be even more opinionated with this respect, although it must be acknowledged that especially for smaller sample sizes or for some improvements, like features with interaction effects, reaching a 100% gold standard quota is also not desirable.

Seventh, interviewees found in-app education about experimentation and hypotheses generation as one of HypGen's key features (received the highest value points). Studies show that lack of expertise is among the top adoption barriers for applying CEx (cf. [20]). With its focus on in-app education, contextual views, and extensive use of explanatory tooltips, HypGen could effectively ingrain this needed expertise into the software product. While perceived as effective, some subjects had some complications learning what HypGen does and how to best use its features. The reason is that many tooltips needed to be read, and topics about statistics or experimentation in context views are often perceived to be quite complicated. An interviewee even suggested moving from text only to graphics or videos that explain certain aspects. So, an effective design may also ingrain expertise into its DSS, but may also focus efforts on user-centric copywriting and improving the understandability of in-app education material.

Eighth, results indicate that developing towards an interactive UI where users like to spend time with is probably an effective design choice for a hypotheses generation DSS. In a similar field, experimentation systems tend to apply a 'UI centric design philosophy' with intuitive templates, graphical roadmap overviews, and many more visual elements as well (cf. [5, 84, 50, 48]). The same is likely true for a hypotheses generation tool. Product Managers mentioned that they overall value HypGen's intuitive workflow design while also highlighting that simplicity and UX could be further improved. Other options such as a command line interface would very likely not speak towards non-technical Product Managers and would make in-app education and context-views much harder to achieve. Future process executions for hypotheses generation will very likely become faster than now ca. 20 minutes. But still, thinking, reviewing suggestions, trying out different experiment configurations, and so forth will consume significant time from Product Managers. Thus, an effective design should provide the visual means that users enjoy working with the system for longer times. Related to this, a target optimization priority for future designers may not be to dramatically speed up the process in order to approach zero. Findings give strong support that time is spent well thinking and engaging

[3] although of a higher rigor as explorative post-launch analyses as reported in the interview study

for a few minutes more. This time is perceived to be spent well for a hopefully valuable hypothesis that leads to a winning experiment and hence a product change that does not need to be rolled back.

### 8.1.3. Implications for Ongoing Projects

Embedding a DSR project in the greater project ecosystem demonstrates the immediate relevance of the knowledge generated. In order to embed the thesis, links to the scientific DSR landscape as well as to practitioner projects are presented.

Ongoing DSR, as well as potential follow-up projects, can benefit from this study's knowledge contributions. Lehtelä aims to explore lightweight CEx solutions in her Ph.D. thesis that prove value for 'companies with few resources or projects involving an early-stage product to benefit from the advantages of CE[x]' [147]. Her considerations of what 'lightweight CEx' likely means can be enriched by this research's presentation of the state of CEx and the problems identified during hypotheses generation at the case study startup. Moreover, leveraging design knowledge generated in the course of pragmatic hypotheses generation, i.e. hypotheses generation that is aware of a startup's certainty-speed trade-off might help her to better solve problems in the resource-constrained environment that is in her research's focus.

Additionally, recent DSR work in the *Hypotheses Engineering* field [52] has designed and evaluated a cognitive mapping technique for better hypotheses elicitation called HyMap. As the project's authors Melegati et al. tend to contribute major articles to the field, their potential follow-up projects may be enriched by HypGen's learnings. Likewise, the research project xCESE (eXtreme Continuous Experimentation in Software Engineering) has presented an early version of their software Undulate [148] in 2023 which aims at supporting CEx from experiment design to analysis. As they describe their software as having the potential to help small to medium-sized companies, their ongoing efforts might be inspired to extend their scope to hypotheses generation before experiment design.

Also, practitioner projects can benefit from the knowledge generated as part of the thesis' DSR project. Other projects seem to follow similar paths than HypGen did and also aim at providing Product Managers with a helping hand for improving their software product. At around the time of HypGen's final evaluations being completed, in July 2023 *beyond.so* [149] has released their growth assistant (cf. Figure 8.1) in a public beta version (cf. release video [150]). Similar to HypGen, their team of SaaS industry experts [4] builds a software based on the integration of event data to compile data-driven insights about improvement opportunities or potential experiments to launch. Their design seems to be currently (as of August 2023) limited to retention data only from the product analytics tool Mixpanel [101] (which includes just the letter R in the HEART framework). This, and the overall impression to be a limited MVP indicates that their growth assistant is currently in an earlier stage than HypGen. Further, it seems that they do not involve the Product Manager as closely in a hypotheses generation workflow nor do they focus that much on in-app education and situative knowledge. Due to the potential benefits of

[4] with former Product Managers and Engineers from Google, N26, Freeletics, or Typeform [151]

the thesis DSR's insights on the growth assistant's design, the author reached out to the development team and invited them to get access to the thesis paper once published.

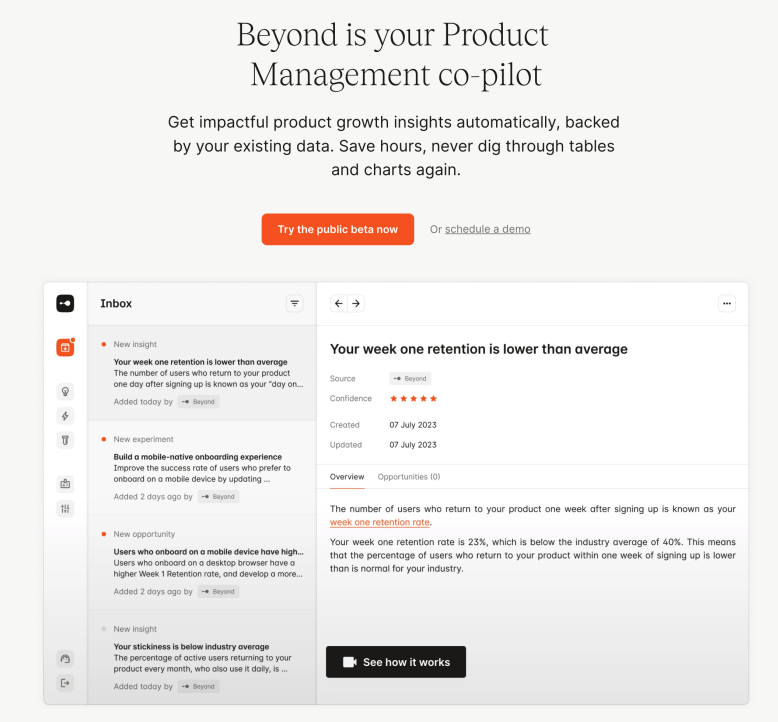



Figure 8.1.: beyond.so Public Beta

## 8.2. Review of Design Science Research Guidelines

In this section, the author evaluates how Hevner et al.'s [1] seven research guidelines for DSR projects are met in the thesis project. The author presents the original guidelines on the left and how this project has addressed them to the right. Table 8.1 shows an adapted version of Hevner et al.'s *Design Science Research Guidelines* [1] enriched by a column 'Satisfaction'. This column gives a brief overview of how this research has addressed high standards in DSR.

| Guideline | Satisfaction |
|---|---|
| Design as an Artifact: *Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation* | The instantiation 'HypGen', a DSS has been produced. It allows Product Managers in a B2B SaaS company to improve their hypotheses generation practices. |
| Problem Relevance: *The objective of design-science research is to develop technology-based solutions to important and relevant business problems* | CEx adoption beyond industry leaders is only moderate, hypotheses generation has not yet received major attention, and there is little technical support for defining hypotheses. Further, the interview study has shown that hypotheses tend to be mistaken for informal assumptions that do not allow for validation in the course of CEx. As hypotheses are often rejected in CEx, understanding hypotheses generation better and improving this discipline is of high interest. |
| Design Evaluation: *The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods* | A rigorous scientific evaluation has been conducted with a mix of quantifiable and open survey questions, observation of task completion behavior, integration of think-aloud, and other UX research methods (interaction plots, mouse heatmaps). Altogether, they are used to address the artifact's evaluation criteria and can yield support for the artifact's satisfaction of addressing the overall research question. |
| Research Contributions: *Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies* | Reviewing the literature yielded both a grounded motivation for the need of the artifact and achieved an integration of existing knowledge into HypGen's design. A variety of knowledge contributions for effective design approaches has been generated, projects that could benefit in the industry have been identified, and contributions to literature have been made. |
| Design Rigor: *Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact* | The interview study has been extensively analyzed using thematic analysis and multiple labeling hierarchies. High standards in application design (e.g. DDD, modularization, interactive web application, modern tech stack, unit/integration test suite) have been ensured. The evaluation benefited also from rigorous methods with a great focus on high-resolution results (e.g. showing individual subject's data points). Dry-runs for interview study and evaluation have been conducted. |
| Design as a Search Process: *The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment* | Explorative research to explore the application environment has been conducted. Multiple build-and-evaluate cycles while integrating feedback from real product management professionals have been executed. Available real-world data (usage event datasets, survey results) as well as working on real use cases with HypGen has ensured a fit of the solution to the real problem environment as closely as possible. |

| | |
|---|---|
| Communication of Research: *Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences* | Results are interpreted and discussed with respect to addressing the application environment's problems as well as contributions to the research landscape which is aimed at inspiring both practitioners and scholars. The design of HypGen is communicated both to management (UI flow) and technology audiences (e.g. UMLs). |

Table 8.1.: Satisfaction of DSR Guidelines. Adapted from [1]

Overall, it can be concluded that this thesis project has addressed DSR guidelines properly. The thesis puts great emphasis on ensuring sound contributions to the knowledge base by applying scientific rigor in exploring the environment and evaluating the artifact. The core research question and artifact evaluation criteria together indicate that the core research question: *'How should a Decision Support System be designed to improve the generation of testable and valuable product improvement hypotheses in a SaaS Start-Up context?'* can be successfully answered with the development of HypGen and its evaluation. However, mostly due to the limited nature of a thesis DSR project, the case study setting, limitations to the research as well as future research opportunities have been identified.

## 8.3. Limitations

As with all research, this DSR project has certain limitations. The author presents threats to validity that might have occurred. Additionally, it describes how the research mitigates potential issues to ensure fair contributions to the body of knowledge.

First, the selection of study participants in combination with the researcher's affiliation with the case study company might result in threats to validity. As the researcher and study subjects (interview subjects, evaluation subjects) worked for the same company during the DSR project, this can have effects on overall validity. For example, the researcher's expectations about desirable outcomes could have been unintentionally communicated to subjects. To mitigate this in the case of the interview study, the researcher has asked open-ended questions. To mitigate biases in the case of the evaluation, the researcher did not actively instruct subjects beyond introducing the overall evaluation - otherwise task completion instructions and surveys have been handed out in textual form.

Second, the moderate sample size of 5 interviewees in the interview study and likewise 5 subjects in the evaluation may decrease internal validity. While 5 interviewees are neither uncommon [141] nor generally bad for qualitative research [152], not all interviewees have reported equally rich insights on each aspect. Hence, generated learnings are sometimes influenced by only a few observations and a higher audience would have been desirable[5]. The author mitigated effects by aiming for deep engagement with the problem domain

[5]however not realistic as the circle of relevant people to talk to in product management is limited at a single company

and artifact evaluation by applying a broad range of different methods with scientific rigor (e.g. labeling, conditional follow-up questions, interaction log analysis, mouse click events, observing behavior, survey) with a high level of detail.

Third, participant attrition and history effects between explorative pre-study and final evaluation may need to be considered. Two out of the five subjects in the interview study did not participate anymore in the evaluation as they had left the company. Risks from history and attrition effects have been mitigated by not explicitly benchmarking results from the evaluation to results of the pre-study. Both the interview study and evaluation study stand on their own and results are interpreted individually without e.g. connecting quotes from the pre-study with individual evaluation results of the evaluation.

Fourth, operationalizations for complex phenomena naturally bring concerns for construct validity with them. When the design artifact is evaluated, evaluation criteria such as 'usefulness', 'learnability' as well as the coverage of the overall research question have been measured with the help of answers to Likert-Scale questions. While standard practice in research, the exact framing of the Likert-Scale question's statement ('In how far do you agree to the statement: ...?') might influence subjects' answers. The researcher has mitigated this issue by testing the questionnaire upfront and asking testers what exactly they understood once they answered each question. Also, the author is blending survey results with qualitative follow-up questions and observations during the evaluation. Further, the author does not interpret the low-sample results as valid significant statistics about all product management professionals but portrays findings as descriptive statistics for the subjects in scope which includes shedding light on individual answers.

Fifth, conformity bias might play a role when interpreting evaluation results. In the interview study, interviewees report having the wish to become more data-driven and idealize the data-driven Product Manager as a role model. Having HypGen as a potential tool in their hands to achieve their desired self (i.e. the data-driven Product Manager), this might over-emphasize their perception of the artifact's impact on their everyday work life. The study mitigated this issue by letting subjects experience the tool with real usage event and survey data. Hence, subjects have been working on realistic product improvements so their perception is likely closer to a future application as opposed to working with hypothetical scenarios.

Sixth, when it comes to external validity, the generalizability of this study to other domains is constrained. With usually greater focus on internal validity in a case study setting, achieving external validity (generalizing results) is comparatively more complex to achieve as a single environment is investigated in-depth. However, once research conducted in the course of case studies is applying rigorous scientific methods, still, causal relationships can be illustrated that are likely applicable beyond the case study's scope [153]. As part of this study, the problems of product management in a B2B SaaS context with a less mature product and adoption problems of engaging in hypotheses generation as well as experimentation have been described. This context is obviously distinct from industry leader examples in the US that are publishing a lot about CEx which diminishes its applicability. However, the research findings of this project fit well into the overall findings on the problem space (cf. [20, 18, 51]). Also, HypGen's design choices seem

to reflect core product ideas that experienced practitioners are starting to pursue (cf. beyond.so's growth assistant [149]). Supporting this, HypGen also received positive sentiment to recommend the DSS to other Product Managers. Taken together, it can be assumed that research findings have some noteworthy degree of applicability in subdomains of the software industry. Applicability is likely higher in B2B SaaS environments with lower experimentation maturity and smaller companies that are working on improving less mature products.

## 8.4. Avenues for Future Research

This research contributes to an improved understanding of decision-making for which product changes to develop in a B2B SaaS product development environment. Doing so, the research sheds light on the underrepresented role of hypotheses and experimentation in this context. Further, the study investigates how an effective solution can be designed to support the generation of statistically testable product hypotheses that are valuable for experimentation. While providing many knowledge contributions, the research's scope was limited. This in turn also results in manifold opportunities for future research.

First, recent advancements in LLMs can be very likely leveraged in the context of hypotheses generation. For instance, there are promising projects like MetaGPT [154] that aim at creating a multi-agent framework for automating many tasks in software development along with automating parts of the Product Manager's work. While this thesis research argues for a larger integration of the human in the loop due to her strengths in domain knowledge and grounding machine-made suggestions to the problem at hand, there is definitely more value to extract from LLMs. The current research only provided ChatGPT [155] generated improvement potentials for dimensions in the HEART framework without taking the context of a feature change into consideration. Fine-tuning of a model to a given company's context including feeding customer requests and so forth into a LLM could add further value to support hypotheses generation.

Second, the study reports that subjects perceive HypGen as having the potential to improve the generation of statistically testable and more valuable hypotheses in the long run. However, to what degree, when, or even if these perceptions turn out to be true would need a long-term observation with a thought-through research design. Such a long-term evaluation would have exceeded the time scope of the Master's Thesis. Yet, this open question motivates subsequent work to be conducted by fellow scholars.

Third, a replication study of HypGen in another environment, if closer (e.g. B2B SaaS at another company) or more distant in the software industry, might yield greater insight into HypGen's applicability. In this study, general applicability is assumed to some degree as findings fit into the research landscape and Product Managers reported the tendency to recommend HypGen to other Product Managers. Testing HypGen (or similar designs incorporating its ideas) in other settings will definitely add further knowledge towards understanding the generation of statistically testable and valuable hypotheses in the course of CEx.

Fourth, comparative shortcomings in HypGen's design speak for future DSR projects

achieving even better results. A better data integration of research artifacts such as personas, job to be done, or qualitative insights and an even more intuitive user interface would potentially extend the quality of a hypotheses generation DSS. Further, thinking about a design that integrates a tool like HypGen directly into either the experimentation platform (as an upfront planning workflow) or in a product analytics tool (as a downstream export workflow) will probably be worthwhile paths to follow.

Fifth, while not part of the capabilities of this prototype, an interviewee suggested an interesting design path for future work. Given collected information about the degree of how the previous hypotheses have improved a certain product area. Then, having knowledge about how effective some previous product changes were (e.g. an average of effect per won experiment) for a product area could also help to prioritize where to focus on next. Such considerations could also give hints about where the product is likely easier to improve, and where it seems to be harder. As experiments can have high failure rates, using the winning likelihood and degree of impact for hypotheses prioritization might be a worthwhile design path to pursue. Future research might concentrate on addressing this potential research topic.

Sixth, this research puts a template-workflow-centered design philosophy into its focus, i.e. sequentially filling out information to achieve a desired outcome. In fact, this is not the only possible approach to address an end-to-end decision-making problem. Other interfaces such as chatbots may help in translating hypotheses even more intuitively into natural language. And it might be worthwhile to help the Product Manager to co-create better hypotheses by other means. Such a tool could guide the Product Manager not in a UI workflow, but in a question-and-answer fashion to generate a hypothesis. Further, a chatbot might play the role of a devil's advocate who is proactively challenging the Product Manager with alternative solutions, potential biases, or different viewpoints to her assumptions.

# Bibliography


[1] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[2] J. Vom Brocke, R. Winter, A. Hevner, and A. Maedche, "Special issue editorial - accumulation and evolution of design knowledge in design science research: a journey through time and space," *Journal of the Association for Information Systems*, vol. 21, no. 3, pp. 520–544, 2020.

[3] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[4] S. Liu, A. H. Duffy, R. I. Whitfield, and I. M. Boyle, "Integration of decision support systems to improve decision support performance," *Knowledge and Information Systems*, vol. 22, pp. 261–286, 2010.

[5] J. Rydberg, "Spotify's new experimentation platform (part 2)," 2020, accessed: 2023-09-15. [Online]. Available: https://engineering.atspotify.com/2020/11/02/spotifys-new-experimentation-platform-part-2/

[6] J. Linowski, "Best checkout ui patterns," 2023, accessed: 2023-09-15. [Online]. Available: https://goodui.org/patterns/screen/checkout/

[7] K. Schwaber and J. Sutherland, "The scrum guide," 2020, accessed: 2023-09-15. [Online]. Available: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

[8] C. Parnin, E. Helms, C. Atlee, H. Boughton, M. Ghattas, A. Glover, J. Holman, J. Micco, B. Murphy, T. Savor *et al.*, "The top 10 adages in continuous deployment," *IEEE Software*, vol. 34, no. 3, pp. 86–95, 2017.

[9] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann, "Online controlled experiments at large scale," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge discovery and Data Mining*, 2013, pp. 1168–1176.

[10] K. Kevic, B. Murphy, L. Williams, and J. Beckmann, "Characterizing experimentation in continuous deployment: A case study on bing," in *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 2017, pp. 123–132.

[11] M. Tingley, W. Zheng, S. Ejdemyr, S. Lane, and C. McFarland, "Decision making at netflix," 2021, accessed: 2023-09-15. [Online]. Available: https://netflixtechblog.com/decision-making-at-netflix-33065fa06481

[12] J. Parks, "Scaling airbnb's experimentation platform," 2017, accessed: 2023-09-15. [Online]. Available: https://medium.com/airbnb-engineering/https-medium-com-jonathan-parks-scaling-erf-23fd17c91166

[13] C. Tang, T. Kooburat, P. Venkatachalam, A. Chander, Z. Wen, A. Narayanan, P. Dowell, and R. Karl, "Holistic configuration management at facebook," in *Proceedings of the 25th Symposium on Operating Systems Principles*, 2015, pp. 328–343.

[14] J. Rydberg, "Spotify's new experimentation platform (part 1)," 2020, accessed: 2023-09-15. [Online]. Available: https://engineering.atspotify.com/2020/10/29/spotifys-new-experimentation-platform-part-1/

[15] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, "Controlled experiments on the web: survey and practical guide," *Data mining and knowledge discovery*, vol. 18, no. 1, pp. 140–181, 2009.

[16] R. Kohavi and R. Longbotham, "Online controlled experiments and a/b testing." *Encyclopedia of Machine Learning and Data Mining*, vol. 7, no. 8, pp. 922–929, 2017.

[17] J. Melegati, X. Wang, and P. Abrahamsson, "Hypotheses engineering: first essential steps of experiment-driven software development," in *2019 IEEE/ACM Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution (RCoSE/DDrEE)*. IEEE, 2019, pp. 16–19.

[18] J. Melegati, E. Guerra, and X. Wang, "Understanding hypotheses engineering in software startups through a gray literature review," *Information and Software Technology*, vol. 133, p. 106465, 2021.

[19] E. Ries, *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Currency, 2011.

[20] G. Schermann, J. Cito, P. Leitner, U. Zdun, and H. C. Gall, "We're doing it live: A multi-method empirical study on continuous experimentation," *Information and Software Technology*, vol. 99, pp. 41–57, 2018.

[21] F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch, "Building blocks for continuous experimentation," in *Proceedings of the 1st international workshop on rapid continuous software engineering*, 2014, pp. 26–35.

[22] D. McKinley, "Design for continuous experimentation: Talk and slides," 2012, accessed: 2023-09-15. [Online]. Available: https://mcfunley.com/design-for-continuous-experimentation

[23] D. I. Mattos, "On experimentation in software-intensive systems," Ph.D. dissertation, Chalmers University of Technology and University of Gothenburg, 2021.

[24] B. Fitzgerald and K.-J. Stol, "Continuous software engineering: A roadmap and agenda," *Journal of Systems and Software*, vol. 123, pp. 176–189, 2017.

[25] V. R. Basili, R. W. Selby, and D. H. Hutchens, "Experimentation in software engineering," *IEEE Transactions on software engineering*, vol. 12, no. 7, pp. 733–743, 1986.

[26] T. Crook, B. Frasca, R. Kohavi, and R. Longbotham, "Seven pitfalls to avoid when running controlled experiments on the web," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge discovery and Data Mining*, 2009, pp. 1105–1114.

[27] H. H. Olsson and J. Bosch, "The hypex model: from opinions to data-driven software development," in *Continuous software engineering*, J. Bosch, Ed. New York City NYC: Springer, 2014, pp. 155–164.

[28] H. H. Olsson, "Challenges and strategies for undertaking continuous experimentation to embedded systems: Industry and research perspectives," *Agile Processes in Software Engineering and Extreme Programming*, vol. 277, 2018.

[29] F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch, "The right model for continuous experimentation," *Journal of Systems and Software*, vol. 123, pp. 292–305, 2017.

[30] G. Schermann, J. Cito, and P. Leitner, "Continuous experimentation: challenges, implementation techniques, and current research," *Ieee Software*, vol. 35, no. 2, pp. 26–31, 2018.

[31] A. Fabijan, P. Dmitriev, H. H. Olsson, and J. Bosch, "The evolution of continuous experimentation in software product development: from data to a data-driven organization at scale," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, pp. 770–780.

[32] ——, "The benefits of controlled experimentation at scale," in *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2017, pp. 18–26.

[33] G. Schermann, "Continuous experimentation for software developers," Ph.D. dissertation, University of Zurich, Deparment of Economics, 2017.

[34] R. Ros and P. Runeson, "Continuous experimentation and a/b testing: A mapping study," in *2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE)*. IEEE, 2018, pp. 35–41.

[35] F. Auer and M. Felderer, "Current state of research on continuous experimentation: a systematic mapping study," in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2018, pp. 335–344.

[36] G. Baxter and I. Sommerville, "Socio-technical systems: From design methods to systems engineering," *Interacting with computers*, vol. 23, no. 1, pp. 4–17, 2011.

[37] S. Huang, "Experimentation platform at zalando: Part 1 - evolution," 2021, accessed: 2023-09-15. [Online]. Available: https://engineering.zalando.com/posts/2021/01/experimentation-platform-part1.html

[38] D. I. Mattos, J. Bosch, and H. H. Olsson, "Multi-armed bandits in the wild: pitfalls and strategies in online experiments," *Information and Software Technology*, vol. 113, pp. 68–81, 2019.

[39] A. Deb, S. Bhattacharya, J. Gu, T. Zhou, E. Feng, and M. Liu, "Under the hood of uber's experimentation platform," 2018, accessed: 2023-09-15. [Online]. Available: https://eng.uber.com/xp/

[40] T. D. Cook, D. T. Campbell, and W. Shadish, *Experimental and quasi-experimental designs for generalized causal inference*. Boston, MA: Houghton Mifflin, 2002.

[41] C. McFarland, M. Pow, and J. Gick, "Quasi experimentation at netflix," 2018, accessed: 2023-09-15. [Online]. Available: https://netflixtechblog.com/quasi-experimentation-at-netflix-566b57d2e362

[42] J. Overgoor, "Experiments at airbnb," 2014, accessed: 2023-09-15. [Online]. Available: https://medium.com/airbnb-engineering/experiments-at-airbnb-e2db3abf39e7

[43] A. Fabijan, H. H. Olsson, and J. Bosch, "Customer feedback and data collection techniques in software r&d: a literature review," in *International Conference of Software Business*. Springer, 2015, pp. 139–153.

[44] P. Hodgson, "Feature toggles (aka feature flags)," 2017, accessed: 2023-09-15. [Online]. Available: https://martinfowler.com/articles/feature-toggles.html

[45] M. Schultzberg, O. Kjellin, and J. Rydberg, "Spotify's new experimentation coordination strategy," 2021, accessed: 2023-09-15. [Online]. Available: https://engineering.atspotify.com/2021/03/10/spotifys-new-experimentation-coordination-strategy/

[46] A. Deng, "Objective bayesian two sample hypothesis testing for online controlled experiments," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 923–928.

[47] Kameleoon, "Frequentist vs bayesian statistics: which method should you use for your a/b tests?" 2023, accessed: 2023-09-15. [Online]. Available: https://www.kameleoon.com/en/blog/ab-testing-bayesian-frequentist-statistics-method

[48] Optimizely, "Welcome to the home of exceptional digital experiences," 2023, accessed: 2023-09-15. [Online]. Available: https://www.optimizely.com/optimize/

[49] Kameleoon, "Experiment your way," 2023, accessed: 2023-09-15. [Online]. Available: https://www.kameleoon.com/en

[50] Posthog, "Posthog the modern platform for product analytics and experimentation," accessed: 2023-09-15. [Online]. Available: https://posthog.com/product

[51] M. Gutbrod, J. Münch, and M. Tichy, "How do software startups approach experimentation? empirical results from a qualitative interview study," in *International Conference on Product-Focused Software Process Improvement*. Springer, 2017, pp. 297–304.

[52] J. Melegati, E. Guerra, and X. Wang, "Hymap: Eliciting hypotheses in early-stage software startups using cognitive mapping," *Information and Software Technology*, vol. 144, p. 106807, 2022.

[53] M. Tingley, W. Zheng, S. Ejdemyr, S. Lane, and C. McFarland, "Interpreting a/b test results: false negatives and power," 2021, accessed: 2023-09-15. [Online]. Available: https://netflixtechblog.com/interpreting-a-b-test-results-false-negatives-and-power-6943995cf3a8

[54] V. Mäntylä, B. Lehtelä, and F. Fagerholm, "The viability of continuous experimentation in early-stage software startups: A descriptive multiple-case study," in *International Conference on Product-Focused Software Process Improvement*. Springer, 2022, pp. 141–156.

[55] H. H. Olsson, H. Alahyari, and J. Bosch, "Climbing the" stairway to heaven"–a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software," in *2012 38th euromicro conference on software engineering and advanced applications*. IEEE, 2012, pp. 392–399.

[56] L. Lucian, "Meet wasabi, an open source a/b testing platform," 2017, accessed: 2023-09-15. [Online]. Available: https://www.intuit.com/blog/innovation/engineering/meet-wasabi-an-open-source-ab-testing-platform

[57] J. vom Brocke, A. Hevner, and A. Maedche, "Introduction to design science research," in *Design science research. Cases*. Springer, 2020, pp. 1–13.

[58] A. R. Hevner, "A three cycle view of design science research," *Scandinavian journal of information systems*, vol. 19, no. 2, p. 4, 2007.

[59] J. O. Johanssen, A. Kleebaum, B. Paech, and B. Bruegge, "Practitioners' eye on continuous software engineering: An interview study," in *Proceedings of the 2018 International Conference on Software and System Process*, 2018, pp. 41–50.

[60] K. L. Barriball and A. While, "Collecting data using a semi-structured interview: a discussion paper," *Journal of Advanced Nursing-Institutional Subscription*, vol. 19, no. 2, pp. 328–335, 1994.

[61] T. P. Johnson, "Snowball sampling: introduction," *Wiley StatsRef: Statistics Reference Online*, 2014.

[62] F. Ghaljaie, M. Naderifar, and H. Goli, "Snowball sampling: A purposeful method of sampling in qualitative research," *Strides in Development of Medical Education*, vol. 14, no. 3, p. 67670, 2017.

[63] R. Hoda, J. Noble, and S. Marshall, "Using grounded theory to study the human aspects of software engineering," in *Human Aspects of Software Engineering*, 2010, pp. 1–2.

[64] LinkedIn, "Linkedin - welcome to your professional community," accessed: 2023-09-15. [Online]. Available: https://www.linkedin.com/

[65] D. Council, "Framework for innovation," accessed: 2023-09-15. [Online]. Available: https://www.designcouncil.org.uk/our-resources/framework-for-innovation/

[66] M. Cagan, "Marty cagan: Twitter account," accessed: 2023-09-15. [Online]. Available: https://twitter.com/cagan

[67] Condens, "condens: Supercharge your ux research analysis," accessed: 2023-09-15. [Online]. Available: https://condens.io/

[68] Productboard, "productboard: The customer-centric product management platform," accessed: 2023-09-15. [Online]. Available: https://www.productboard.com/

[69] Pendo, "pendo.io: Software that makes your software better," accessed: 2023-09-15. [Online]. Available: https://www.pendo.io/

[70] Tableau, "tableau: Find easy answers to hard questions," accessed: 2023-09-15. [Online]. Available: https://www.tableau.com/

[71] Miro, "miro: Where hybrid teams get work done," accessed: 2023-09-15. [Online]. Available: https://miro.com/index/

[72] Figma, "figma: Nothing great is made alone," accessed: 2023-09-15. [Online]. Available: https://www.figma.com/

[73] Maze, "Maze: Build amazing products. without the guesswork." accessed: 2023-09-15. [Online]. Available: https://maze.co/

[74] Jira, "Jira software: The no. 1 software development tool used by agile teams," accessed: 2023-09-15. [Online]. Available: https://www.atlassian.com/software/jira

[75] Confluence, "Confluence software: Accomplish more together," accessed: 2023-09-15. [Online]. Available: https://www.atlassian.com/software/confluence

[76] Workpath, "Workpath: The enterprise software solution for effective strategy implementation," accessed: 2023-09-15. [Online]. Available: https://www.workpath.com/en/home

[77] K. Popper, *The Logic of Scientific Discovery (first published 1935)*. New York City NYC: Routledge, 2005.

[78] R. H. Bonczek, C. W. Holsapple, and A. B. Whinston, *Foundations of decision support systems*. New York City NYC: Academic Press, 1981.

[79] H. A. Simon, *The new science of management decision*. New York City NYC: Harper & Brothers, 1960.

[80] G. P. Huber, *Managerial Decision Making*, ser. Management Applications Series. Illinois: Glenview, 1980.

[81] T. Singer, "Coming soon: Confidence — an experimentation platform from spotify," 2023, accessed: 2023-09-15. [Online]. Available: https://engineering.atspotify.com/2023/08/coming-soon-confidence-an-experimentation-platform-from-spotify/

[82] S. Berreiter, "Lessons from 15+ saas companies and their continuous experimentation platforms," 2021, acccessed: 2023-09-15. [Online]. Available: https://medium.com/@berreiter.stefan/lessons-from-15-saas-companies-and-their-continuous-experimentation-platforms-4e9d1911e437

[83] D. Ryabov, "Twitter experimentation: technical overview," 2015, accessed: 2023-09-15. [Online]. Available: https://blog.twitter.com/engineering/en_us/a/2015/twitter-experimentation-technical-overview

[84] S. Xiang, "Building pinterest's a/b testing platform," 2016, accessed: 2023-09-15. [Online]. Available: https://medium.com/pinterest-engineering/building-pinterests-a-b-testing-platform-ab4934ace9f4

[85] R. Atachiants, "Reliable and scalable feature toggles and a/b testing sdk at grab," 2018, accessed: 2023-09-15. [Online]. Available: https://engineering.grab.com/feature-toggles-ab-testing

[86] A. Ivaniuk and J. Liu, "Making the linkedin experimentation engine 20x faster," 2020, accessed: 2023-09-15. [Online]. Available: https://engineering.linkedin.com/blog/2020/making-the-linkedin-experimentation-engine-20x-faster

[87] D. Ryabov, "The what and why of product experimentation at twitter," 2015, accessed: 2023-09-15. [Online]. Available: https://blog.twitter.com/engineering/en_us/a/2015/the-what-and-why-of-product-experimentation-at-twitter-0

[88] H. H. Olsson and J. Bosch, "From opinions to data-driven software r&d: A multi-case study on how to close the 'open loop' problem," in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, 2014, pp. 9–16.

[89] J. Bosch, H. H. Olsson, J. Björk, and J. Ljungblad, "The early stage software startup development model: a framework for operationalizing lean principles in software startups," in *International Conference on Lean Enterprise Software and Systems*. Springer, 2013, pp. 1–15.

[90] H. H. Olsson and J. Bosch, "Towards continuous customer validation: A conceptual model for combining qualitative customer feedback with quantitative customer observation," in *International Conference of Software Business*. Springer, 2015, pp. 154–166.

[91] Optimizely, "Design an effective hypothesis," accessed: 2023-09-15. [Online]. Available: https://support.optimizely.com/hc/en-us/articles/4410282998541-Design-an-effective-hypothesis

[92] Productplan, "Ice scoring model," accessed: 2023-09-15. [Online]. Available: https://www.productplan.com/glossary/ice-scoring-model/

[93] Conversion, "Pie prioritization framework," accessed: 2023-09-15. [Online]. Available: https://conversion.com/framework/pie-framework/

[94] Atlassian, "What every product manager needs to know about product analytics," accessed: 2023-09-15. [Online]. Available: https://www.atlassian.com/agile/product-management/product-analytics

[95] Mixpanel, "Popular topics what is product analytics," accessed: 2023-09-15. [Online]. Available: https://mixpanel.com/blog/what-is-product-analytics/

[96] T. Mäkilä, A. Järvi, M. Rönkkö, and J. Nissilä, "How to define software-as-a-service–an empirical study of finnish saas providers," in *International Conference of Software Business*. Springer, 2010, pp. 115–124.

[97] J. Burby, A. Brown, W. S. Committee *et al.*, "Web analytics definitions," *Web Analytics Association*, vol. 1, no. 1, pp. 1–34, 2007.

[98] Pendo, "Pendo Developer Documentation track events," accessed: 2023-09-15. [Online]. Available: https://developers.pendo.io/docs/?bash#track-events

[99] B. J. Jansen, "Understanding user-web interactions via web analytics," *Synthesis lectures on information concepts, retrieval, and services*, vol. 1, no. 1, pp. 1–102, 2009.

[100] Mixpanel, "Tracking FAQs creating a tracking plan," accessed: 2023-09-15. [Online]. Available: https://help.mixpanel.com/hc/en-us/articles/115004519886-Creating-a-Tracking-Plan

[101] ——, "Build Better Products powerful, self-serve product analytics to help you convert, engage, and retain more users." accessed: 2023-09-15. [Online]. Available: https://mixpanel.com/home/

[102] Amplitude, "No. 1 product analytics," accessed: 2023-09-15. [Online]. Available: https://amplitude.com/

[103] Mixpanel, "Mixpanel Retention Analysis retain to grow," accessed: 2023-09-15. [Online]. Available: https://mixpanel.com/solutions/retain/

[104] Amplitude, "Amplitude Insight smart alerting for product monitoring," accessed: 2023-09-15. [Online]. Available: https://amplitude.com/insight

[105] S. Gupta, A. Leszkiewicz, V. Kumar, T. Bijmolt, and D. Potapov, "Digital analytics: Modeling for insights and new methods," *Journal of Interactive Marketing*, vol. 51, no. 1, pp. 26–43, 2020.

[106] D. Chaffey and M. Patron, "From web analytics to digital marketing optimization: Increasing the commercial value of digital analytics," *Journal of Direct, Data and Digital Marketing Practice*, vol. 14, no. 1, pp. 30–45, 2012.

[107] J. R. Saura, P. Palos-Sánchez, and L. M. Cerdá Suárez, "Understanding the digital marketing environment with kpis and web analytics," *Future Internet*, vol. 9, no. 4, p. 76, 2017.

[108] D. McClure, "Startup Metrics For Pirates: AARRR!!! startup metrics for product marketing and product management," accessed: 2023-09-15. [Online]. Available: https://www.slideshare.net/dmc500hats/startup-metrics-for-pirates-long-version?ref=https://product-frameworks.com

[109] K. Rodden, H. Hutchinson, and X. Fu, "Measuring the user experience on a large scale: user-centered metrics for web applications," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2010, pp. 2395–2398.

[110] GoodUI, "GoodUI.org learn from what leading companies a/b test," accessed: 2023-09-15. [Online]. Available: https://goodui.org/

[111] P. B. Kruchten, "The 4+1 view model of architecture," *IEEE software*, vol. 12, no. 6, pp. 42–50, 1995.

[112] tiangolo/fastapi, "Fastapi framework, high performance, easy to learn, fast to code, ready for production," 2023, accessed: 2023-09-15. [Online]. Available: https://fastapi.tiangolo.com/

[113] pandas.pydata.org, "pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the python programming language," 2023, accessed: 2023-09-15. [Online]. Available: https://pandas.pydata.org/

[114] SQLAlchemy, "The python sql toolkit and object relational mapper," 2023, accessed: 2023-09-15. [Online]. Available: https://www.sqlalchemy.org/

[115] encode/uvicorn, "Uvicorn is an asgi web server implementation for python," 2023, accessed: 2023-09-15. [Online]. Available: https://www.uvicorn.org/

[116] pydantic/pydantic, "Pydantic is the most widely used data validation library for python." 2023, accessed: 2023-09-15. [Online]. Available: https://docs.pydantic.dev/latest/

[117] plotly, "Dash is the original low-code framework for rapidly building data apps in python," 2023, accessed: 2023-09-15. [Online]. Available: https://dash.plotly.com/

[118] opensource.faculty.ai, "Dash bootstrap components," 2023, accessed: 2023-09-15. [Online]. Available: https://dash-bootstrap-components.opensource.faculty.ai/

[119] postgresql.org, "postgres - the world's most advanced open source relational database," 2023, accessed: 2023-09-15. [Online]. Available: https://www.postgresql.org/

[120] drawio, "Security-first diagramming for teams," 2023, accessed: 2023-09-15. [Online]. Available: https://www.drawio.com/

[121] docker.com, "Docker: Develop faster. run anywhere." 2023, accessed: 2023-09-15. [Online]. Available: https://www.docker.com/

[122] docs.pytest.org, "pytest: helps you write better programs," 2023, accessed: 2023-09-15. [Online]. Available: https://docs.pytest.org/en/7.4.x/

[123] SAIT, "Ieee citation style: Images/videos," 2023, accessed: 2023-09-15. [Online]. Available: https://libguides.sait.ca/IEEE/imagesvideos

[124] python.org, "Python 3.10.0," 2023, accessed: 2023-09-15. [Online]. Available: https://www.python.org/downloads/release/python-3100/

[125] docs.python.org, "dataclasses — data classes," 2023, accessed: 2023-09-15. [Online]. Available: https://docs.python.org/3/library/dataclasses.html

[126] plantUML, "Plantuml in a nutshell," 2023, accessed: 2023-09-15. [Online]. Available: https://plantuml.com/

[127] GNU, "Gnu make," 2023, accessed: 2023-09-15. [Online]. Available: https://www.gnu.org/software/make/manual/make.html

[128] virtualenv.pypa.io, "Virtualenv," 2023, accessed: 2023-09-15. [Online]. Available: https://virtualenv.pypa.io/en/latest/

[129] E. Evans, *Domain-driven design: tackling complexity in the heart of software*. Boston MA: Addison-Wesley Professional, 2004.

[130] D.-h. Shin, "Python domain-driven-design(ddd) example," 2023, accessed: 2023-09-15. [Online]. Available: https://github.com/qu3vipon/python-ddd/tree/main

[131] N. Gonçalves, D. Faustino, A. R. Silva, and M. Portela, "Monolith modularization towards microservices: Refactoring and performance trade-offs," in *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2021, pp. 1–8.

[132] M. Fowler, "Cqrs," 2011, accessed: 2023-09-15. [Online]. Available: https://martinfowler.com/bliki/CQRS.html

[133] H. Feng, "Handy functions for a/b testing in python," 2019, accessed: 2023-09-15. [Online]. Available: https://henryfeng.medium.com/handy-functions-for-a-b-testing-in-python-f6fdff892a90

[134] J. Diong, "Calculating sample size for a paired t-test," 2019, accessed: 2023-09-15. [Online]. Available: https://scientificallysound.org/2017/08/03/calculating-sample-size-for-a-paired-t-test/

[135] A. Hevner, N. Prat, I. Comyn-Wattiau, and J. Akoka, "A pragmatic approach for identifying and managing design science research goals and evaluation criteria," in *AIS SIGPrag Pre-ICIS workshop on" Practice-based Design and Innovation of Digital Artifacts"*, 2018.

[136] K. Peffers, M. Rothenberger, T. Tuunanen, and R. Vaezi, "Design science research evaluation," in *Design Science Research in Information Systems. Advances in Theory and Practice: 7th International Conference, DESRIST 2012, Las Vegas, NV, USA, May 14-15, 2012. Proceedings 7*. Springer, 2012, pp. 398–410.

[137] T. J. Ellis and Y. Levy, "A guide for novice researchers: Design and development research methods," in *Proceedings of Informing Science & IT Education Conference (InSITE)*, vol. 10, no. 10. Italy, Cassino, 2010, pp. 107–117.

[138] Y. Sun and P. B. Kantor, "Cross-evaluation: A new model for information system evaluation," *Journal of the American Society for Information Science and Technology*, vol. 57, no. 5, pp. 614–628, 2006.

[139] J. Venable, J. Pries-Heje, and R. Baskerville, "Feds: a framework for evaluation in design science research," *European journal of information systems*, vol. 25, pp. 77–89, 2016.

[140] N. Prat, I. Comyn-Wattiau, and J. Akoka, "A taxonomy of evaluation methods for information systems artifacts," *Journal of Management Information Systems*, vol. 32, no. 3, pp. 229–267, 2015.

[141] C. Carroll, P. Marsden, P. Soden, E. Naylor, J. New, and T. Dornan, "Involving users in the design and usability evaluation of a clinical decision support system," *Computer methods and programs in biomedicine*, vol. 69, no. 2, pp. 123–135, 2002.

[142] C.-K. Yang and C. Wacharamanotham, "Alpscarf: Augmenting scarf plots for exploring temporal gaze patterns," in *Extended abstracts of the 2018 CHI conference on human factors in computing systems*, 2018, pp. 1–6.

[143] C. M. Christensen, T. Hall, K. Dillon, and D. S. Duncan, "Know your customers' jobs to be done," *Harvard Business Review*, vol. 94, no. 9, pp. 54–62, 2016.

[144] E. Miller, "How many subjects are needed for an a/b test?" 2023, accessed: 2023-09-15. [Online]. Available: https://www.evanmiller.org/ab-testing/sample-size.html

[145] N. K. Dhand and M. S. Khatkar, "Statulator: An online statistical calculator. sample size calculator for comparing two independent means." 2014, accessed: 2023-09-15. [Online]. Available: https://www.evanmiller.org/ab-testing/sample-size.html

[146] F. Reichheld, *The ultimate question: Driving good profits and true growth.* Boston, MA: Harvard Business School Press, 2006.

[147] B. Lehtelä, "Exploring the essence of continuous experimentation: Minimum prerequisites for light-weight ce," *ACM SIGSOFT Software Engineering Notes*, vol. 48, no. 1, pp. 49–55, 2023.

[148] T. Asikainen, T. Männistö, and E. Huovila, "Undulate: Extreme continuous experimentation of separately deployable software," *SoftwareX*, vol. 22, p. 101400, 2023.

[149] beyond.so, "beyond.so - your growth assistant," 2023, accessed: 2023-09-15. [Online]. Available: https://beyond.so/

[150] S. Neubauer, "Lightning demo of beyond beta product," 2023, accessed: 2023-09-15. [Online]. Available: https://www.loom.com/share/676ae61c90a94a63a2dfb8b4fa249464?sid=c8c5de9b-9c8e-4846-8da3-1bedab727fbe

[151] beyond.so, "beyond.so - about us," 2023, accessed: 2023-09-15. [Online]. Available: https://beyond.so/about

[152] Nielsen-Research, "Why you only need to test with 5 users," 2000, accessed: 2023-09-15. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[153] N. Cartwright, *Internal and External Validity Issues in Case Study Research*, ser. Strategies for Social Inquiry. Cambridge University Press, 2022, p. 27–116.

[154] MetaGPT, "Metagpt: The multi-agent framework," accessed: 2023-09-15. [Online]. Available: https://github.com/geekan/MetaGPT/blob/main/README.md

[155] OpenAI, "chatgpt," 2023, accessed: 2023-09-15. [Online]. Available: https://chat.openai.com/

[156] S. Berreiter, "Stop making bad product decisions — start applying continuous experimentation," 2021, accessed: 2023-09-15. [Online]. Available: https://medium.com/@berreiter.stefan/how-continuous-experimentation-makes-you-to-stop-making-bad-product-decisions-every-day-a40831dafe28

[157] Conversion, "Better results with evidence," accessed: 2023-09-15. [Online]. Available: https://conversion.com/

[158] U. Zdun, R. Capilla, H. Tran, and O. Zimmermann, "Sustainable architectural design decisions," *IEEE software*, vol. 30, no. 6, pp. 46–53, 2013.

# Acronyms

**ADD** Architectural Design Decisions. 67, 144

**AoI** Area of Interest. 87, 94, 98, 100, 148

**API** Application Programming Interface. 50, 68–70, 77–79

**B2B** Business to Business. iii, v, vi, 2, 12, 13, 37, 54, 55, 105, 106, 112, 114, 115, 142, 144, 146

**BC** Bounded Context. 68–71, 73, 74, 76, 78, 79

**CEx** Continuous Experimentation. iii, v, vi, 1, 3–8, 11, 12, 39–42, 45, 55, 75, 83, 85, 91, 105–110, 112, 114, 115, 136, 144

**CI/CD** Continuous Integration and Continuous Delivery/Deployment. 3

**CQRS** Command Query Responsibility Separation. 70

**CRUD** Create-Update-Read-Destroy. 71

**DDD** Domain-Driven Design. 68, 70, 72, 112, 144

**DSR** Design Science Research. i, iii, v, xi, 1, 2, 9–13, 46, 51, 55, 63, 68, 83–85, 91, 103, 105, 106, 110, 111, 113, 115, 135, 144

**DSS** Decision Support System. iii, v, 1, 2, 12, 39, 40, 42–44, 46, 49, 50, 54, 65, 67, 69, 80, 83, 85, 89, 103, 107–109, 112, 113, 115, 116, 144

**GC** Goodness Criteria. 42, 85

**HEART** (H)appiness (E)ngagement (A)doption (R)etention (T)ask Success. 52–54, 58, 60, 63, 67, 72, 79, 87, 96, 99, 107, 108, 110, 115, 144, 151

**IDSS** Intelligent Decision Support System. xiii, 44

**LLM** Large Language Model. 54, 108, 115

**MDE** Minimum Detectable Effect. 7, 39, 47, 60, 61, 90, 100, 101

**MVF** Minimum Viable Feature. 22

*Acronyms*

**MVP** Minimum Viable Product. 29, 110

**NPS** Net Promoter Score. 85, 91, 103

**REST** Representational State Transfer. 69, 70, 77–79

**SaaS** Software as a Service. iii, v, vi, 2, 5, 8, 12, 15, 18, 19, 37, 39, 50, 54, 55, 57, 69, 73, 83–85, 103, 105, 107, 110, 112–115, 144, 146, 151

**UI** User Interface. 22, 29, 33, 44, 68, 75, 77, 79, 81, 94, 97, 109, 113, 116, 142–144

**UX** User Experience. 17, 18, 24, 27, 28, 31, 33, 34, 64, 85, 86, 91, 109, 112, 142

# A. Appendix

## A.1. Implementation Project

The project can be found in the respective GitLab repository at: `https://gitlab.swa.univie.ac.at/cx/master_berreiter`. Here, the reader can find a general high-level overview of the code project with the most important information displayed. A more in-depth description can be found in the GitLab repository's *README.md* file.

**High-level folder structure**

- **evaluation/...**: Jupyter notebooks to analyze the evaluation
- **implementation**/: HypGen implementation
  - **source_data/...**: Event and survey data (both demo and real-world data)
  - **src/backend**/: Fastapi backend, split up in modules
    - **experimentation/...**:
    - **product_admin/...**:
    - **product_analytics/...**:
    - **shared_kernel/...**:
    - **suggestions/...**:
  - **tests/backend/...**: Unit and integration tests for backend
  - **tracking/...**: Service for recording mouse locations and UI events
  - **ui/...**: Implementation of frontend with Dash
  - **.env**: Credentials, ...
  - **docker-compose.yaml**: For spinning up databases
  - **fastapi_app.py**: Root file for the backend, aggregating all BCs
  - **Makefile**: Handy commands to execute different services
  - **requirements.txt**: Dependencies for fastapi app
- **iterations**/: Documentation of intermediate designs and feedback
- **README.md**: Documentation of the code project

## A.2. Author's Affiliation with the Case Study Company

The researcher has worked during the Design Science Research (DSR) project at the case study company. Once it became apparent that HypGen will build on the company's data, the researcher received access to collect event and survey data from the company's product analytics system for demonstration purposes. As a further benefit, the researcher had easier access to product management professionals of the company for pre-study and evaluation, whereby participation in the study has been entirely voluntary.

The development of HypGen and the DSR project in general was not part of the researcher's scope of work at the company. Also, the researcher did not receive payment from the company or related entities in order to conduct the research. Also, no existing code snippets of the company have been incorporated into HypGen's code.

The researcher prevents biases by not communicating any information that was not relevant to the pre-study interviewees nor to the evaluators, besides obviously running the interview or evaluations. By design, both the pre-study questions were kept open and the evaluation was conducted as autonomously as possible. This helps to mitigate affiliation related intervention problems.

## A.3. Author's Blogging about CEx

The author has started blogging about Continuous Experimentation on the platform medium.com. The rationale was to create interest in the community on research in CEx and to collect feedback what might inspire the audience.

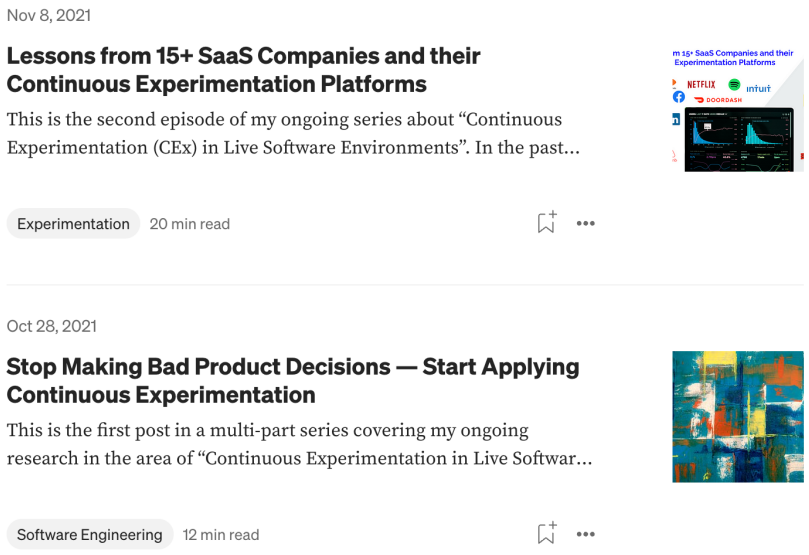



Figure A.1.: Medium Articles on Experimentation

The thesis author has contributed two articles, one on CEx in general [156] and the other on CEx platforms [82] which both received attention in the practitioner community. Taken together, the two published articles received over 1,300 views until August 2023. Also, the content has been recognized by the influential practitioner and scholar Ronny Kohavi[1] as well as was featured in an industry newsletter on experimentation (from Widerfunnel, now Conversion [157]). Parts of the articles have been bookmarked by readers which yielded interesting insights while whole articles have been also added to reading lists. The themes of reading lists where mostly collections of data-driven product management articles.

The articles do not include any information on the core of the DSR project, e.g. the project's research question, the interview study, the development of HypGen or its evaluation.

[1] Received a clap from Ronny Kohavi's medium account

## A.4. Dry Runs & Support

### A.4.1. Pre-Study Dry Run

The author has made a dry-run with a former Product Manager. It has been the same setting as the real interviews. This means, the interviewee got a briefing beforehand, the interview was conducted and recorded with the same tool. The only differences were that after the interview, the author and test interviewee had a short discussion regarding which questions have been unclear and what might need to be redesigned. Also during the interview, the test interviewee gave short feedback once he did not understand a certain aspect. The interview has not been transcribed nor labeled as it was just used to improve the quality of the interview itself, not the analysis thereof.

### A.4.2. Pre-Study External Labeler

The external labeler (a distinct person than the test interviewee) was provided with one out of the five interviews in a setting where he was not aware of labels which have been used by the author. The labeler was aware of the research objective as well as the research questions which influenced his labeling of relevant information. When comparing results on the training sample, one can observe that both labelers share major commonalities in how they relate information to labels and to research question. Both labelers used multiple labels if one piece of information has been both e.g. a process step (RQ1) whilst also giving information on the concrete artifact (RQ2) which is produced. Overall, most labelings have been overlapping. Referring to overlapping means that the content of the labelled text have been labelled conceptually the same. The label itself might be with a different wording, but describes the same phenomenon. Comparing the length of the labeled text per label results in equally the same patterns between both labelers.

With regard to differences, the external labeler instinctively created a hierarchy of labels (e.g. a label group 'Process' and corresponding sub-labels like Process>Conceptualization, Process>Investigation) whereas the author used 'flat labelings'. Further, the external labeler additionally labeled stakeholders which are mentioned during the interview (internal, external). In terms of the labeling of rules and heuristics, the external labeler showed more awareness for explicit statements which state that something is not a rule, which makes in itself a rule again. On the other hand, the author used labeling categories which the external labeller did not use such as perceptions of the participants (e.g. wishes, frustrations, ...). Moreover, the author provided short descriptions of labels which mark larger text snippets to condense it's key aspect.

After discussing the commonalities and differences, the author and external labeler agreed to continue the labeling to account for both labeling approaches' special advantages. First, the author opted to restructure his labels according to a labelling hierarchy. Second, the labeling of stakeholders has been added to the authors set of labels and text was labelled accordingly. Third, the author also included 'non-rules' in the labeling of rules. Finally, adding short descriptions to longer labeled text snippets has been made a clear necessity for the further labelling by the author alone.

### A.4.3. Evaluation Dry Run

The evaluation has been conducted in a dry run with two business informatics students with product management knowledge. The motivation has been to test the technical set-up, understandability of tasks, and wordings of questionnaire questions. Overall, the dry run has been successful. The evaluation and completing the tasks with HypGen worked as expected. Further, the technical set-up worked as expected (i.e. mouse interaction monitoring, user interaction logs, screen recording). Still, feedback led to smaller adjustments in the questionnaire as some questions were found to not be self-explanatory or in other cases overly specific.

## A.5. Semi-Structured Interview Guideline

The now following interview guideline served as guidance to answer the four exploration research questions. Worth noting, RQ3 has been asked before RQ2 as it was found during the dry-run that it might be easier to get valuable answers when first asking about rules and heuristics, and then asking about artifacts and concepts. Also, information on RQ4 (which tools product management is using) was collected throughout the interview. In the end it was controlled in a question where subjects are asked to list all tools. This was the point where potentially missing tools could be identified.

### A.5.1. Onboarding

I already pre-filled some information. Can you give me a quick sparring if I have entered the basic demographic information correctly?

- What is your basic demographic information (name, age, gender, nationality, degree)?
- What is your relevant career path in the scope of product management (stations, roles, duration)?
- How would you describe your current role in software development in a compact way?

### A.5.2. First Block

In this block, I like to find out: *What are the different steps in product management in the course of deciding on a concrete feature to develop?*

- Imagine one exemplary feature which has been developed in the past months: Can you guide me through the process (not the technical details) until you ultimately arrived at the shipped feature - starting from the earliest beginning you can think of?
  - Which of these key activities follow each after in a sequential logical?
  - Which activities are rather continuous in nature?
  - Which activities are building backlinks (creating a cycle in the process)?
- Which activities are you perceiving as the ones where you are primarily involved?
- Is this exemplary scenario covering the vast majority of present scenarios, or are there clear deviations from this scenario of developing a feature?
  - If so, can you explain why these scenarios are varying?
  - If so, can you explain what the key derivations in the process and your involvement in the process are?

### A.5.3. Second Block

In this block, I like to find out: *What is the role of rules, heuristics, and intuition in the process?* The goal is to create a better understanding of your tasks and how your thinking process.

- In these key activities you just named, can you tell me which information you consult, research or create in order to guide your decision making in these areas?
- Can you think of activities in this process where you are applying - explicit or implicit - rules?
  - If explicit can you tell or show me the concrete rule?
  - If implicit, a) can you tell me about this implicit rule you are applying? and b) can you tell the reason this rule is implicit and not explicit?
- Can you think of activities in this process where rather intuition and experience is of importance?
  - If so, a) can you point to key activities where this is dominating, and b) can you go through one exemplary situation where you are factoring in your intuition and experience?

### A.5.4. Third Block

In this block, I want to find out: *Which concepts and artifacts is product management referring to, researching, or creating in the course of the decision-making process?*

- In the process of your key activities, which information artifacts play a role and to which key activity do they belong? E.g. an artifact may be a customer story written down (or in your head) or a feature description in a software tool)
- In the process of your key activities, which of the artifacts are you a) creating new or b) transforming?
  - Per information artifact: Can you take me to a small example and tell what you do?
  - Per information artifact: Can you give an indication how (e.g. time, resource, intellect, ...) extensive it is to create this artifact.
  - Per information artifact: If no resource constraints exist, would this artifact differ from today?
  - If so, can you tell me what might differ and what your rational behind this is?
- Are you currently missing some artifacts?
  - If so, could you tell what you are missing?
  - If not, what makes you come to this conclusion?

- Are you using terms such as assumptions, hypotheses, features (and more) in your development process?
  - If so, can you provide a short definition for each term?
  - If so, after having defined all terms, which one's can you delineate more clearly and which ones might be rather ambiguous / overlapping?

### A.5.5. Control Questions

In this block, we want to cover some secondary aspects which may or may not have an impact on the previous answers. Also, it aims to ask for tools which might have not already been mentioned before, so it addresses the question: 'Which tools are supporting product management in their daily work which play a role in decision-making about product improvements?'

- Are there any authors or frameworks which have heavily influenced your thought process?
  - If so, can you tell which ones and which core ideas you got inspired by?
  - If so, how are you making their guidance work for you?
- Which (analogous, digital) tools are you using in your daily work which play a role in the topics we covered today?
- Do you want to highlight the most important functionality which they are covering for you?
  - If you are missing something in these tools, what would that be (in short)?

### A.5.6. Outro

Thanking the participant for her participation. Give a short outlook on the further process. Room for participant to be in the loop (get updates) and to take part in the final evaluation of the prototype.

## A.6. Roles Mentioned in Decision-Making Process

| | Role | Type | Role Description |
|---|---|---|---|
| 1 | Product Manager | internal | main decision-maker on product roadmap, which problems to solve, which features to develop and in which sequence, collaborating in solution brainstorming, conducting research interviews, analyzing quantitative usage data to draw conclusions |
| 2 | UX Researcher | internal | conducting qualitative interviews and condensing product relevant insights for decision making, testing prototypes with real users |
| 3 | Product Designer | internal | collaborating in brainstorming, developing UI designs and prototypes |
| 4 | Software Developer | internal | co-creating concept with technical know-how, developing and releasing the software |
| 5 | CEO / Leadership | internal | inspires the product strategy and rough roadmap, contributes feature requests and forwards customer feedback |
| 6 | Customer Success Manager | internal | forwarding customer feedback to product management and quasi-validating assumptions |
| 7 | Sales Manager | internal | forwards customer feedback and inspires feature ideas through feedback of successful or failed customer pitches |
| 8 | Competitor | external | company that building a product with a set of features who target at solving the same or adjacent problems for the same or similar customers |
| 9 | Customer | external | company that is purchasing B2B licenses to use the software for its employees, the users |
| 10 | User | external | users are either using or not using the software to certain extents, potentially provide feature feedback-/requests |

Table A.1.: Roles Within The Product Management Process

## A.7. Product Manager Tool Landscape

| | Tool | Type | Short Description |
|---|---|---|---|
| 1 | productboard [68] | knowledge management | organizing various forms of customer insights |
| 2 | condens [67] | interview research | interview recordings, transcripts and content labels |
| 3 | pendo [69] | data analytics | session analytics based on web requests |
| 4 | tableau [70] | data analytics | inhouse analytics based on production database |
| 5 | miro [71] | collaborative canvas tool | unstructured post-its and comments used for multiple purposes |
| 6 | figma [72] | creating (clickable) designs | UI designs and clickable user flows |
| 7 | maze [73] | user testing | testing clickable prototypes and getting primarly qualitative feedback |
| 8 | jira [74] | define development tickets | definition of epics, user stories and estimation thereof |
| 9 | confluence [75] | general documentation | documenting features, decisions, ... |
| 10 | workpath [76] | strategic planning | documenting mid-term product priorities and their progress |

Table A.2.: Current Product Manager Tool Landscape

## A.8. Main Architectural Design Decisions

To document Architectural Design Decisions (ADD) that have significantly influenced the design of HypGen, the author has opted to leverage Zdun et al.'s (wh)y-statement approach[158]. To allow for lean documentations, they have proposed to document ADD in the following pattern: *'In the context of <use case/user story u>, facing <concern c> we decided for <option o> to achieve <quality q>, accepting <downside d>'.*

**fastapi backend (python):** In the context of *building backend apis for HypGen*, facing *the need for flexibility and speed in an explorative and data-analysis centric DSR project*, the author has decided *on a fastapi backend written in python* to achieve *fast implementation speed while also having easy access to data science libraries*, accepting that *relying on python has the downside of opting in a dynamically typed language.*

**plotly Dash frontend (python):** In the context of *offering an intuitive DSS UI to Product Managers*, facing *the need for a highly interactive user design that also displays graphs*, the author has decided *on a plotly dash frontend written in python (an abstraction of React.js with focus on data UIs)* to achieve *state-of-the-art UIs while also being able to leverage python data science code if necessary*, accepting *that the support for a classic frontend framework may be more advanced.*

**Domain-Driven Design (DDD) and storing domain aggregates:** In the context of *embedding the complex domain knowledge about hypotheses generation in the artifact's code base*, facing *the urge to focus above all on the domain and not technical implementation details*, the author has decided *to apply a layered DDD where each subsystem/module is storing its domain aggregates* to achieve *domain driven – not data storage entric – design*, accepting *that this limits the benefits from using ORMs and leads to more boilerplate code.*

**HEART framework for product health analysis:** In the context of *having an overarching framework to relate individual product metrics to*, facing *the need to cover a B2B SaaS product (area) holistically*, the author has decided *to apply Google's HEART framework* to achieve *broad coverage of potential product health issues in a B2B software product which can be identified with the help of product metrics*, accepting *that this might lead to less applicability of HypGen for non B2B SaaS environments where other frameworks (e.g. Pirate framework) would be preferred.*

**One metric per HEART dimension:** In the context of *analyzing product area health to uncover improvement opportunities*, facing *the need to validate if such a framework-driven health analysis approach works out*, the author has decided *to offer one highly configurable metric per HEART dimension* to achieve *some degree of flexibility for product metrics while staying within a manageable complexity scope*, accepting *that evaluation subjects may perceive it as too minimal.*

**Frequentist statistics:** In the context of *applying statistics for hypotheses testing in CEx*, facing *the need to reason about a hypothesis' likely feasibility given an experimentation set-up with statistical properties and a given target metric*, the author has decided *to apply frequentist and not bayesian statistics* to achieve *the possibility to use sample size estimations*, accepting *that bayesian approaches are seen to offer lower experimentation durations.*

**Real-world data loaded from .csv files:** In the context of *incorporating real-world usage behavior and survey data in HypGen*, facing *the need to manage technical complexity and data protection*, the author has decided *to load dumps of event and survey data (.gitignored) and to not communicate with the product analytics system of the company* to achieve *working with the same data while offering the same experience in the evaluation but not on the cost of complexity/security*, accepting *that it decreases the production-readiness of HypGen.*

**PostHog as Experimentation System:** In the context of *integrating HypGen with a third-party experimentation system*, facing *the need to showcase the general workings*, the author has decided *to integrate with PostHog (free-tier for experimentation, open source code)* to achieve *the desired demonstration purpose of an experimentation integration*, accepting *that other platforms have more market shares which would decrease HypGen's applicability in a production-ready setting.*

## A.9. Final Evaluation Questions

| | Category | Question | Scale |
|---|---|---|---|
| Q1 | ex ante | How often are you defining changes to your software product (e.g. features) in the form of statistically testable hypotheses? | 7pt Likert: (never) 1 to 7 (always) |
| Q2 | ex ante | How often are you applying statistical testing of product hypotheses with the help of real post-rollout usage data? | 7pt Likert: (never) 1 to 7 (always) |
| Q3 | ex ante | Please mark if you have ever conducted one of these experimentation approaches? Options: A/B Test or Randomised Controlled Experiment; Pre-Post Test; Multi-Armed Bandit; Segmentation Test | for each option: yes/no |
| Q4 | ex ante | Which scenarios can you imagine in which you would hire HypGen to support your work as a SaaS Product Manager? | open question |
| Q5 | ex post | In how far do you agree with the following statement: This tool has the potential to support that product improvements are more likely translated into testable hypotheses? | 5pt Likert: (strongly disagree) -2 to +2 (strongly agree) |
| Q6 | ex post | In how far do you agree with the following statement: This tool has the potential to help with the generation of more valuable product hypotheses? | 5pt Likert: (strongly disagree) -2 to +2 (strongly agree) |
| Q7 | ex post | Applicability: How likely is it that you recommend a software leveraging HypGen's design to a Product Manager at a similar-sized software company? | (very unlikely) 1 to 10 (very likely) |
| Q8 | ex post | Business Alignment: In how far do you agree with the following statement: This tool can contribute to greater adoption of continuous experimentation practices in B2B SaaS environments? | 5pt Likert: (strongly disagree) -2 to +2 (strongly agree) |
| Q9 | ex post | Completeness: In how far do you agree with the following statement: This tool is supporting the vast majority of areas which are important when it comes to defining effective, testable Product Improvement hypotheses? | 5pt Likert: (strongly disagree) -2 to +2 (strongly agree) |
| Q10 | ex post | Fit into IT landscape: In how far do you agree with the following statement: The tool fits well into the current tool suite that is used for various tasks in the domain of Product Management? | 5pt Likert: (strongly disagree) -2 to +2 (strongly agree) |
| Q11 | ex post | Learnability: In how far do you agree with the following statement: The tool is designed in a way so that it is easy to learn what it does and how to best use its features? | 5pt Likert: (strongly disagree) -2 to +2 (strongly agree) |
| Q12 | ex post | Simplicity: In how far do you agree with the following statement: This tool is easy to understand and to get desired actions done? | 5pt Likert: (strongly disagree) -2 to +2 (strongly agree) |

| | | | |
|---|---|---|---|
| Q13 | ex post | Traceability: In how far do you agree with the following statement: This tool creates transparency and traceability from business opportunity to released software functionality? | 5pt Likert: (strongly disagree) -2 to +2 (strongly agree) |
| Q14 | ex post | Usefulness: In how far do you agree with the following statement: This tool is useful in the course of defining effective, testable product improvement hypotheses? | 5pt Likert: (strongly disagree) -2 to +2 (strongly agree) |
| Q15 | ex post | Capabilities: How useful do you rate each of these selected HypGen capabilities with respect to adding value to your workflow when defining impactful, testable product improvements? | 5pt Likert: (irrelevant) -2 to +2 (useful) for each of 13 capabilities |
| Q16 - Q21 | ex post | Overall, how difficult or easy did you find the task: <task>? Asks the same question for all 5 tasks, whereby splitting task 4 into two parts (hypothesis definition; experimentation selection) | 5pt Likert: (very difficult) 1 to 7 (very easy) for each of the six questions |
| Q22 | off-boarding | What comes to your mind after having explored the Product Management Decision Support System HypGen? | open question |
| Q23 | off-boarding | What else would you like to mention? | open question |

Table A.3.: Evaluation Survey Questions

## A.10. Areas of Interest Overview

In total, HypGen is tracked extensively with user interaction events. Once a user hovers a tooltip, clicks on a button, inputs text or selects a dropdown item, the specific user event is tracked. As some interactions logically belong to each other, the author uses User Experience Research's idea of Area of Interest. AoI are a collection of user interaction event types whereby one interaction event type can just belong to one of many AoI. An AoI can theoretically be located on multiple pages although this has been seldom the case. From ca. 100 event types tracked, 19 AoI can be aggregated to enrich further understanding. The 19 AoI are presented with screenshots of HypGen to easily connect AoI and functionality.

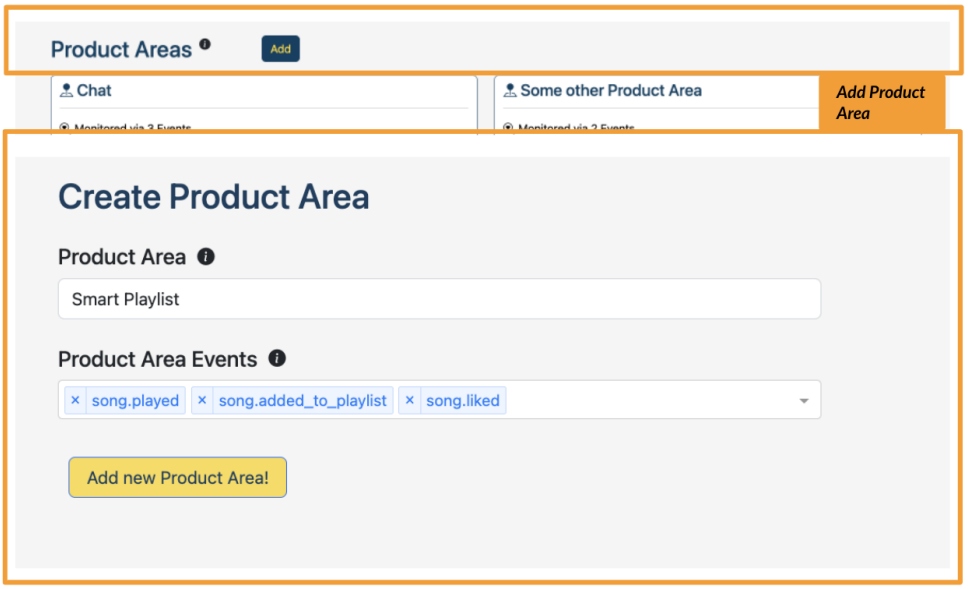



Figure A.2.: AoIs - Product Area

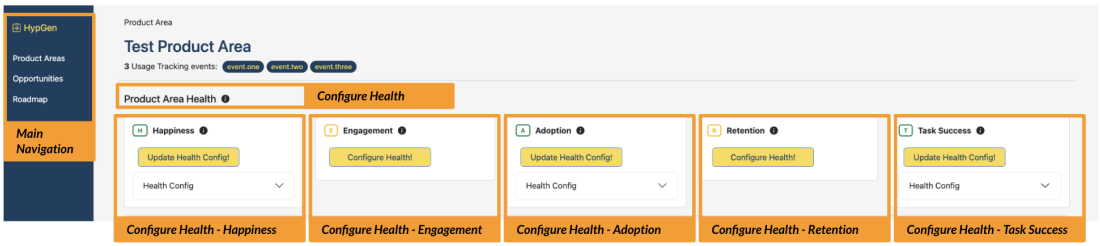



Figure A.3.: AoIs - Product Health (HEART)

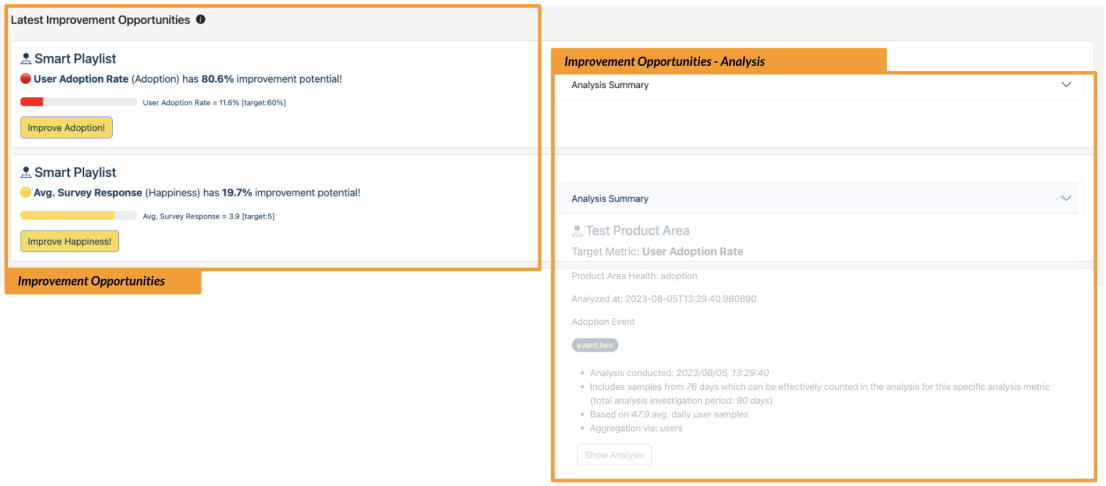



Figure A.4.: AoIs - Improvement Opportunities

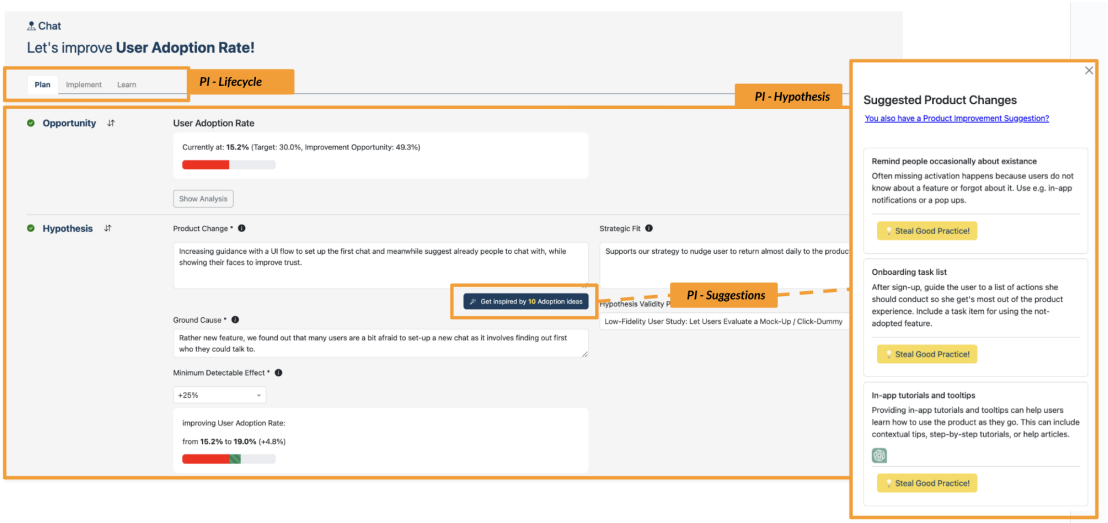



Figure A.5.: AoIs - Hypothesis and Suggestions

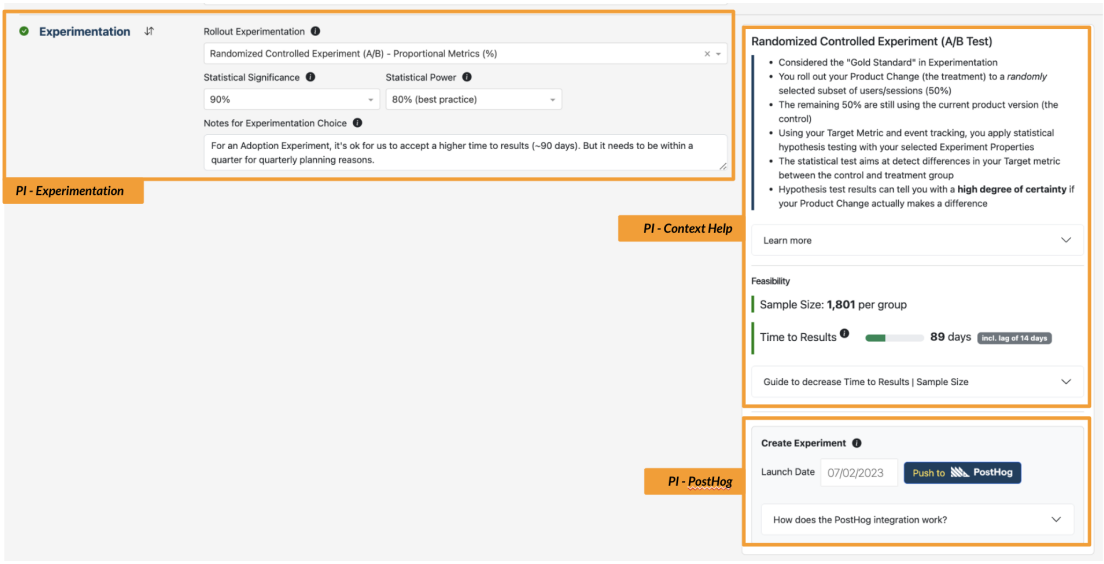



Figure A.6.: AoIs - Experimentation

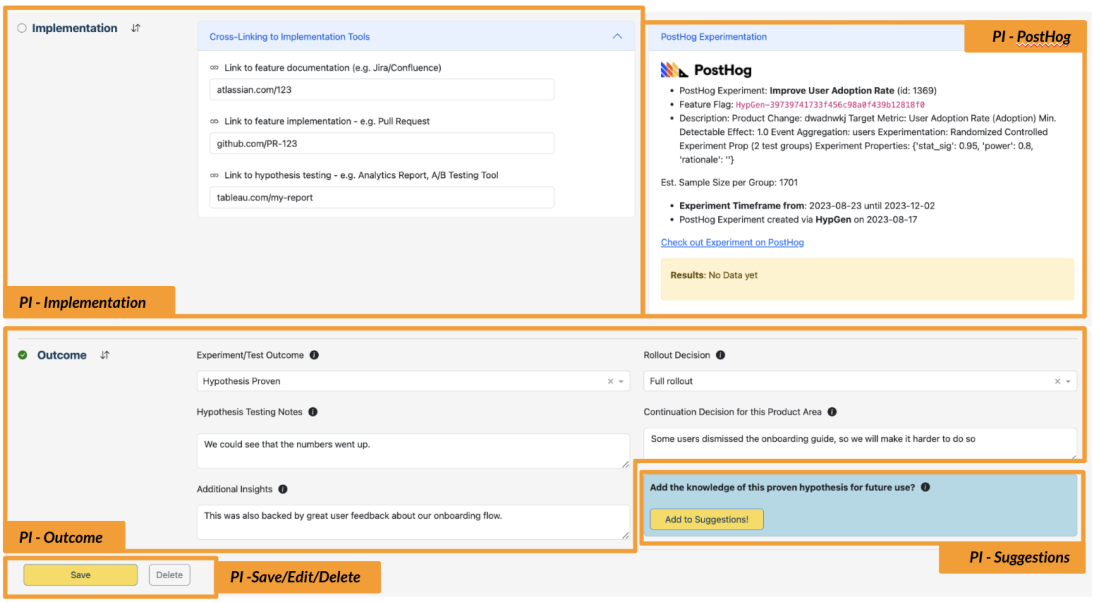



Figure A.7.: AoIs - Hypothesis Implementation & Outcome

## A.11. Event and Survey Data

HypGen builds its data analysis for HEART metric calculation on either event logs (recorded actions that users did with the SaaS product) or survey responses (recorded sentiment of users towards certain features). Below, examples of this data are presented. Events are usually found to have further properties (e.g. properties about the user or the device used, etc.). As they were beyond the scope of the Master's Thesis, they have been omitted. Exemplary data sets for both usage event and survey data can be found in the GitLab project.

### Event

```
{
  "user_id": "fc367acf0dd2c3ad7c4c74523171fe38", # anonymized
  "timestamp": "2020-08-18 23:59:01.000000",
  "event": "project.created",
}
```

Listing A.1: Event Example (HypGen)

### Survey

```
{
  "user_id": "fc367acf0dd2c3ad7c4c74523171fe38", # anonymized
  "timestamp": "2023-03-30 12:19:12",
  "response": 1,
  "survey": "UX Survey - Reading Blogs",
}
```

Listing A.2: Survey Response Example (HypGen)