A Behavioral Design Framework for Optimizing User Behavior: The NUDGE Framework

MASTER THESIS

Johanna Schlinger

Submitted on 30 January 2025

Friedrich-Alexander-Universität Erlangen-Nürnberg
Faculty of Engineering, Department Computer Science
Professorship for Open Source Software

Supervisor:
Georg Schwarz, M.Sc.
Prof. Dr. Dirk Riehle, M.B.A.

Friedrich-Alexander-Universität
Faculty of Engineering

Declaration of Originality

I confirm that the submitted thesis is original work and was written by me without further assistance. Appropriate credit has been given where reference has been made to the work of others. The thesis was not examined before, nor has it been published. The submitted electronic version of the thesis matches the printed version.

Erlangen, 30 January 2025

License

Erlangen, 30 January 2025

Acknowledgements

I would like to thank my supervisor, Georg, for his support and for bringing a sense of calm into the writing process.

Additionally, I am grateful to all the participants who evaluated the NUDGE framework with me - your feedback ultimately shaped it into what I aspired it to be.

I would also like to acknowledge ChatGPT for helping me overcome my deficits in grammar and vocabulary as a non-native speaker.

Finally, special thanks to my partner and friends who encouraged me throughout the whole process and proofread this work.

Abstract

Creating intuitive, user-centric solutions is challenging, particularly in developer-focused fields with limited exposure to behavioral science and usability principles. This thesis introduces the NUDGE Framework, a conceptual behavioral design framework aiming to bridge the gap between design and development. It provides developers — especially those without a design background — with a systematic approach to improving user experience through behavioral design principles. Built on BJ Fogg's Behavior Model and persuasive technology strategies, the framework connects developer goals to relevant behavioral techniques, helping identify design obstacles and apply goal oriented strategies.

A case study within the open source collaboration domain, using the JValue Hub, demonstrates how the framework addresses common usability challenges. The application of the NUDGE Framework to the JValue Hub illustrates its potential to streamline collaboration, enhance user engagement, and improve usability for both novice and experienced users.

An evaluation using a Think-Aloud protocol asserts the framework's accessibility, usability, and effectiveness. Developers who initially expressed skepticism ultimately created unique, context-specific solutions using the framework. These findings suggest the NUDGE Framework equips developers with a practical tool to implement user-centered improvements, fostering user engagement and inclusivity.

Contents

List of Figures

List of Tables

1 Introduction

Great applications do more than just perform well on a technical level; they guide users effortlessly toward their goals. These applications excel not only in their own functionality but also in empowering users to succeed, creating a seamless synergy between the tool and its user. Behavioral design is an emerging discipline that combines principles from behavioral science and design to influence human behavior effectively (Bay Brix Nielsen et al., 2024). While many other design fields focus on making things visually appealing, behavioral design shifts the focus to influencing user behavior - not making things pretty, but making them work for their intended user groups.

Despite its potential to enhance user experience and engagement, incorporating behavioral design principles remains a substantial challenge in developer-focused fields (Bay Brix Nielsen et al., 2024). Many software development firms prioritize technical functionality over user-centric design, often leading to solutions that may function well but fail to resonate with users (Ashley and Desmond, 2005).

This gap is particularly pronounced in open source collaboration environments, where usability is frequently overshadowed by a focus on code and functionality (Levesque, 2004; Raza and Capretz, 2015). In these contexts, developers often lack the tools and guidance necessary to address usability issues systematically. While there is an abundance of behavioral design literature and design principles available, these resources are often fragmented, theoretical, or tailored for experienced designers, making them inaccessible to developers without a design background (Bay Brix Nielsen et al., 2024). This disconnect can leave many usability challenges unaddressed, limiting the potential for truly user-centered solutions.

To address this gap, this thesis introduces a framework to nudge users with behavioral design for goal driven engagement, called the NUDGE framework. NUDGE is a conceptual behavioral design framework designed to bridge the gap between design and development. Built on BJ Fogg's Behavior Model and persuasive technology strategies (Fogg, 2003, 2009), the framework offers a structured approach that enables developers to identify usability challenges, map them to relevant

behavioral principles, and implement actionable solutions. By translating complex behavioral design concepts into accessible and practical tools, the NUDGE Framework aims to empower developers to integrate user-centric design strategies into their workflows without requiring extensive expertise.

The framework is demonstrated through its application to the JValue Hub, a collaboration tool for open-data pipelines. By adapting and applying the NUDGE Framework to this specific context, the thesis illustrates how the framework can address common usability challenges, improve user engagement, and foster a more intuitive collaboration process. The evaluation of the framework, conducted using a Think-Aloud protocol with developers, highlights its accessibility, usability, and potential to drive behaviorally informed design improvements.

2 Problem Definition

The Relevance of Behavioral Design in Development When applied, behavioral design plays a pivotal role in the creation of successful apps as it can directly impact how users engage, interact, and remain loyal to digital products (Voorheis et al., 2022). By focusing on behavioral design, app developers can craft experiences that intuitively guide users, reducing friction and minimizing cognitive load. This approach helps address common challenges such as user engagement, motivation, and the formation of positive behaviors (Fogg, 2003, 2009). For example, well-designed apps can leverage behavioral triggers and feedback loops to keep users engaged over time, enhancing long-term commitment and satisfaction. Moreover, a behavioral design perspective enables developers to anticipate and address potential barriers in the user journey, ensuring that the app's functionality aligns with real human needs and tendencies (Bay Brix Nielsen et al., 2024; Voorheis et al., 2022). In a competitive market, where the difference between a widely adopted app and an abandoned one often hinges on subtle user experience details, incorporating behavioral design principles can be the deciding factor for success.

Balancing Technical Functionality with User-Centered Design According to van Kuijk et al., 2019, there are three main drivers of usability in products. The first is that user-centered design and usability have to be prioritized by the developer or the development team (van Kuijk et al., 2019). Hence, in settings, in which the focus often falls heavily on technical functionality rather than user experience this prioritization is shifted away from design a lot. Developers may lack the training, interest, or motivation to incorporate design principles into their work. This can result in apps that are technically sound but fail to engage users or address their needs effectively (van Kuijk et al., 2019). Secondly, the developers of the product need to have the knowledge to be able to successfully create a user-centered design. However, merely having the knowledge about usability or design practices isn't sufficient for creating a user-centered design. It's also essential that developers have the ability to apply this knowledge, they need to have the freedom and motivation to act on that knowledge, and not only have the right resources but also enough time to do so (van Kuijk et al., 2019).

Without dedicated prioritization of design but only of technical functionality of a product, crucial aspects like intuitive navigation, user motivation, and behavioral triggers might get overlooked. This gap could for example lead to higher drop-off rates, lower user satisfaction, and reduced engagement - challenges that might have been avoided if a user centered design was integrated from the start.

Challenges Faced by Newcomers Entering the Design Space There is a wealth of research-based design principles available that can be used to improve designs and guide decision-making. However, many of these principles are created with designers in mind—targeting individuals who have a background in, or at least a foundational understanding of design. As a result, these principles are often not easily accessible, understandable, or usable for developers or those without prior exposure to design concepts. To make matters more challenging, most discussions and information on methodologies of behavioral design are highly conceptual and fragmented rather than practical or hands-on (Bay Brix Nielsen et al., 2024). This abstract nature might leave developers with the need to interpret and adapt high-level theories to their specific context, which can be time-consuming and inconclusive. Additionally, the challenge isn't just in finding design techniques or principles themselves, but rather in an earlier step: identifying the core problem and mapping the appropriate research or principles to address it. Even after this, the difficulty lies in determining the correct steps to translate the research into actionable strategies that are effective for the specific use case. As a result, developers without a dedicated design background might find it overwhelming to navigate this landscape and translate these principles into actionable strategies for app development. This lack of accessible, practical resources creates an entry barrier for those who want to apply behavioral design effectively, leading to a missed opportunity for creating apps that truly resonate with users. The disconnect between theory and practice makes it difficult to seamlessly integrate behavioral design into the app development workflow (Chiang et al., 2021).

Behavioral Design in Open Source Collaboration This challenge seems to be even more pronounced in open source collaboration, a field heavily dominated by developers and which typically focuses more on technical solutions than on design (Levesque, 2004; Raza and Capretz, 2015). Additionally, the collaboration processes in open source projects are inherently complex, requiring coordination among diverse contributors (Pal et al., 2024). Making these processes intuitive demands thoughtful behavioral design, yet in developer-driven environments, usability and user experience often take a backseat to code and functionality. As a result, many open source platforms suffer from usability issues that make it hard for newcomers to get involved, limiting contributions and hindering project success (Levesque, 2004; Raza and Capretz, 2015; Steinmacher et al., 2015).

Newcomers to open source projects, in particular, face numerous challenges. These include social barriers, such as integrating with established communities, as well as technical obstacles like navigating unfamiliar code bases (Steinmacher et al., 2015). There are also hurdles related to documentation, understanding the existing structure of a project, and finding a suitable starting point for contributions. According to a systematic literature review by Steinmacher et al. (2015) on barriers in open source software, many of these obstacles revolve around effectively conveying information, which comes back to insufficient design, making it hard for new contributors to engage productively (Steinmacher et al., 2015).

Research by Heltweg and Riehle (2023) on open collaborative data engineering highlights further complications in the lack of standardized processes, methods, and tools in these environments that creates prominent challenges. Participants are frequently confronted with new, non-standardized processes, which can be confusing and slow down progress (Heltweg and Riehle, 2023). In this context, having systems that are intuitively understandable would be crucial.

3 Objective Definition

The lack of standardized processes and user-friendly interfaces in open source collaborative data engineering creates notable barriers for both experienced developers and newcomers. Specifically open source environments often suffer from inconsistent practices and a lack of focus on usability, making them challenging to navigate. When standards are absent, a way to counteract this and improve the user experience is to make the tools or processes as intuitively understandable as possible (Heltweg and Riehle, 2023; Levesque, 2004; Raza and Capretz, 2015).

Objective Overview The primary objective of this thesis is to develop a tool that facilitates and simplifies the integration of behavioral design principles into the design of applications and processes, specifically targeting developers with little to no prior experience in design or behavioral design. The tool should offer an intuitive entry point and an accessible introduction into behavioral science concepts, lowering the barrier to entry for those unfamiliar with the field.

Rather than providing general information or abstract concepts, the tool should be adaptable and customizable to specific use cases. It should help developers identify areas with potential for improvement, present options for enhancement, and offer background explanations grounded in behavioral design. The focus should be on proposing practical solutions that are relevant to the user's context, mapping these suggestions directly to identified problems while also considering the developer's goals and intentions.

The tool should aim to serve as a central hub, providing a collection of strategies, simplified approaches, and example scenarios for applying those strategies. It should act as a toolbox, not only for sparking ideas for change but also for guiding users in identifying and exploring the right questions to ask — often the more challenging task. Additionally, the tool should offer clear pathways for discovering effective answers and solutions to those questions.

Targeted Applicability in the Open Source Context The tool should be particularly aimed at supporting the open source collaboration ecosystem, with a focus on workflows and processes common in this space. It is intended to

be relevant to those who use, develop, or contribute to open source tools and processes. While open source projects are the primary context for the tool's application, it should be designed to be flexible enough for broader use.

Existing research by Steinmacher et al., 2015 suggests that many challenges in open source collaboration are not technical but rather related to communication and the effective dissemination of information (Steinmacher et al., 2015). This tool seeks to address such challenges by applying behavioral design principles to improve how information is presented and acted upon in these environments.

The tool should include a curated set of strategies and examples, organized to facilitate problem-solving, and assist developers in identifying the appropriate strategies for their use cases. It should structure the information in a way that distills complex concepts into manageable categories, providing developers with a systematic approach to assessing and improving usability. Thereby, developers should be able to use this tool to iterate on their designs, refine their intentions, and gain a deeper understanding of their users' behavior.

Derived from these objectives, it is essential for the framework to demonstrate strong performance in

- usability,
- effectiveness,
- accessibility, and
- adaptability

to achieve its intended purpose.

4 Solution Design

This chapter outlines the framework's design requirements and structure. To develop and design the framework, we first define our design requirements, derived from the objectives outlined in chapter 3. We then cover how the framework is built, and the practical methods for adapting and applying it to specific contexts. Finally, we reflect on the strengths and limitations inherent in the framework's design and it's alignment with the design requirements defined in chapter 4.1. A more comprehensive evaluation of the framework as a tool, focusing on whether it meets our defined objectives of chapter 3 will be presented in the evaluation chapter 6.

This chapter adopts the following terminology:

- **Users** are the individuals who interact directly with the resulting artifact that is designed or modified using the framework. They are the ones whose behavior is aimed to be influenced.
- **Developers** are those who work with the framework to implement design changes.

These distinctions should avoid confusion between the two kinds of users throughout the chapter.

4.1 Design Requirements

In this chapter, we outline the essential design requirements that guide our framework's development, based on the objectives we set in chapter 3.

Our functional requirements define what the framework must achieve (Chopra et al., 2010), focusing on addressing usability challenges and promoting behavioral design based solutions. These requirements ensure that the framework delivers actionable strategies that developers can easily integrate into their workflows.

Non-functional requirements, such as scalability and simplicity, shape how the framework delivers these strategies (Chung et al., 2000). By prioritizing user-

friendly presentation and adaptability, the framework aims to become accessible to developers of all backgrounds.

The emphasis on behavioral design alignment ensures that the framework itself serves as a practical demonstration of its principles. This approach not only aims to build credibility but also to enhance the framework's usability by embodying the very concepts it seeks to promote.

Functional Requirements We aim for the conceptual framework to address the following core functionalities:

- **Problem Definition:** Help developers recognize and define usability challenges within their projects.
- **Solution Mapping:** Provide pathways for developers to connect specific problems to behavioral design principles.
- **Implementation Support:** Offer practical, context-specific steps that developers can take to improve usability.

These functionalities are meant to ensure that the framework provides a systematic approach to improving usability while remaining grounded in practical, real-world applications.

Non-Functional Requirements To also ensure its usability and effectiveness, the conceptual framework should adhere to several non-functional requirements:

- **Scalability:** The framework should work across projects of varying sizes and complexities, from small tools to large-scale systems.
- **Flexibility:** It must accommodate developers with diverse levels of expertise, from novices to seasoned professionals, without sacrificing depth or utility.
- **Simplicity and Accessibility:** Information must be presented in clear, structured formats, reducing cognitive load and facilitating ease of use.
- **Behavioral Design Alignment:** The framework should exemplify the principles it promotes, providing a consistent and intuitive user experience.
- **Inclusivity in Open Source Contexts:** The framework should address the specific challenges of open source collaboration, such as diverse user groups, non-standardized processes, and limited design focus.
- **Reusability:** It should allow developers to apply the framework across multiple iterations, ensuring sustained usability improvements.

- **Transferability:** The framework should not only address open source specific challenges, but be applicable to different domains and contexts.

These non-functional requirements are meant to ensure that the framework remains practical, adaptable, and user-friendly, even in complex or evolving project environments.

4.2 Theoretical Foundations

What is Behavioral Design? Behavioral design differs from design by combining conventional design and design thinking practices with methodologies from the field of behavioral science (Bay Brix Nielsen et al., 2024; P. J. Cash et al., 2017). So, unlike traditional design disciplines that prioritize aesthetic appeal only, behavioral design seeks to systematically understand and shape behaviors within specific contexts (Bay Brix Nielsen et al., 2024).

Behavioral design, therefore, benefits from the integration of methodologies from behavioral science as well as design thinking. While behavioral science contributes evidence-based methods for understanding and modifying behavior, design thinking emphasizes user-centered, iterative development processes (Bay Brix Nielsen et al., 2024; Tyler, 1966; Wölbling et al., 2012).

Behavioral sciences refer to a range of interdisciplinary fields which study human behavior (Tyler, 1966). Behavioral scientists find methods which diagnose and solve behavioral problems, and doing this through designing behavior changing interventions is known as behavioral design (Bay Brix Nielsen et al., 2024; Bucher, 2020; Voorheis et al., 2022).

Behavioral designers therefore aim to create clear and effective solutions based on the premise that a solution's effectiveness is achieved through influencing user behavior (Bucher, 2020; P. Cash et al., 2022; Voorheis et al., 2022).

To accomplish this, behavioral design must focus on two aspects simultaneously: the desired behavior change, such as a user noticing a specific interface element or drivers reducing their speeding, and the design intervention that facilitates this change, such as emphasizing a UI element visually or installing speed bumps (Khadilkar and Cash, 2020).

According to the Stanford Behavior Design Lab [1], behavioral design is about understanding human behavior and creating designs that facilitate specific changes in it. Its goal is to develop solutions that promote positive behavior change (Stanford Behavior Design Lab, 2019). To support behavioral designers in achieving such positive behavior change, Dr. BJ Fogg from the Stanford Behavior Design

[1] https://behaviordesign.stanford.edu

Lab has developed systematic tools. A key aspect of Fogg's work includes his persuasive technology strategies (Fogg, 2003).

Persuasive Technology Persuasive technology, as defined by Fogg (2003), refers to interactive systems or products designed to influence users' attitudes and behaviors. These technologies aim to make desired actions or outcomes easier to achieve by shaping user behavior in a systematic way. Fogg's foundational work in this field has been instrumental in identifying how technology can be employed to subtly alter human behavior through a set of carefully crafted strategies (Fogg, 2003; Khaled et al., 2005).

The core purpose of persuasive technology is to simplify or modify user actions, such that they are more likely to achieve a specific goal. Behavioral design frequently draws on several strategies to achieve these outcomes, which are grounded in simplifying the process, guiding user decisions, or leveraging user motivation to foster specific behaviors. These strategies are critical for understanding how technology can be used in behavioral change (Khaled et al., 2005; Toledo et al., 2018).

Persuasive Technology Strategies In his work, Fogg identified seven strategies commonly employed in persuasive technology tools (Fogg, 2003). These strategies identified to drive user behavior have further been refined and altered to fit the use case of persuasive, computerized systems by Wildeboer et al. (2016). Wildeboer et al. (2016) refined the selection of techniques within that context to the principles of reduction, tunneling, tailoring, personalization, self-monitoring, simulation, and rehearsal.

4.3 Conceptual Design Process

For the conceptual design process of the framework we started with a comprehensive literature review. This review identified and analyzed critical behavioral design principles, exploring their application and contextual relevance. The objective was to gain an overview of the field and understand how these principles are conveyed and how they are meant to be utilized in practice effectively. Since the framework's usability is of central relevance we employed an iterative development process to refine the framework. This involved cycles of evaluating the framework's usability and based on the evaluation's feedback refining it further. To ensure comprehensive testing, we used a polar sampling approach to pick out the test scenarios, capturing a diverse range of scenarios within the open source collaboration process. A more detailed explanation for this can be found in the demonstration chapter 5.

Our initial focus for the use of the framework was on a specific use case, the

open source collaboration workflow. We selected this domain due to its unique challenges, such as developer-driven environments that often neglect usability and behavioral design in exchange for a stronger focus on technology (Levesque, 2004; Raza and Capretz, 2015). Open source workflows provided a fertile ground for the usage of a framework for behavioral design implementation, as usability improvements in this context could enhance the domain's goal of collaboration and reduce barriers for contributors.

We aimed to transition from a use-case-specific design to a generalized framework. Similar to the method of Explanation-Based Generalization (Mitchell et al., 1986), which leverages domain-specific knowledge to derive general concepts from specific examples by explaining why the example fits the concept, we incorporated the domain-specific knowledge we earlier acquired through our comprehensive literature review to generalize the framework and its usage into a non-use-case specific, more generalized form.

Generalizations are usually based on observing a phenomenon in multiple contexts. In this instance, we generalized the framework without such additional observations, leaving validation of the effectiveness of such generalization to future work.

Application and Adaptation of the Use Case In the upcoming chapters, the open source collaboration workflow use case is used as an illustrative example to show how the framework could be applied in practice and provides a basis for the framework's evaluation. For creating the framework, we therefore combined research-based behavioral insights, iterative development, and a straightforward structure to create a conceptual framework to nudge users with behavioral design for goal driven engagement, called the NUDGE framework.

4.4 The Framework's Conceptual Design

The conceptual framework is structured to facilitate its use through a clear, two-phase process. In the first phase, the framework is adapted to a specific use case. This ensures that it aligns with the unique requirements and context of that particular scenario. In the second phase, this tailored version is applied to the artifact within the specific context, enabling the framework to address the use case effectively. Since nudging users into a specific direction is highly use case and context-sensitive, it's very important to provide very context-specific advice. Hence, the framework's use is divided into 2 different phases: the adaptation phase and the application phase. In the adaptation phase, the framework itself is customized by the developer and adapted to the use case in which the framework is to be used. In the application phase, the behavioral design techniques pointed to by the adapted framework are applied to the artifact itself.

4.4.1 The Framework's Structure: How It Works

The NUDGE framework is structured across three interconnected dimensions: the process dimension, the technique dimension, and the developer intents.

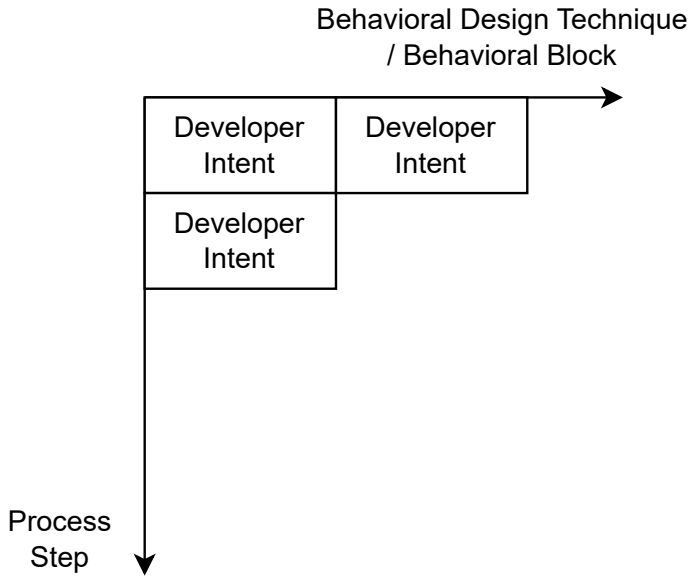


Figure 4.1: The framework's structure

The process dimension encompasses an entire process or subprocess, broken down into individual steps. The technique dimension provides a set of behavioral design techniques derived from Fogg (2003) and Wildeboer et al. (2016) that can be applied within these steps. Developer intents link the two, representing the underlying goals of each process step and guiding the selection of appropriate techniques. Together, these dimensions function by mapping a behavioral design technique to a process step through the intent driving that step. Each dimension will be detailed in the following sections.

The Process Dimension The process dimension portrays the process or user workflow which the developer wants to optimize through the framework. It should contain all steps taken by a user throughout the entire process and therefore each step which the user undergoes on their journey through the evaluated application.

Therefore, the process dimension consists precisely of the process to be optimized and must fully encompass it. The boundaries of the dimensional space should align with the boundaries of the process to be optimized. No steps from a pre-

ceding or subsequent process should be included. This is due to the fact that we designed the framework in a way that it not only optimizes individual user interactions with the application, but also provides the opportunity to optimize the according process as a whole on a conceptual level. To define the process dimension's value space, one creates a process flow-like diagram that pictures all steps a user undergoes throughout the chosen process.

Additionally, one or multiple specific user roles should be defined for the user on hand. Because all behavioral techniques which are used later in the process need to be fitted well to the specific user group, taking the user's abilities into consideration is specifically important here.

Therefore, the process dimension consists of an ordered sequence of well-defined process steps that altogether make up the process that is to be optimized or which contains the usability issues that are to be eradicated with the usage of the NUDGE framework as well as the according user roles.

We will provide more detailed information on how one should define, size and align each process step in the form of a set of rules of thumb in chapter 4.5.1. For now, understanding the basic concept of what the process dimension portrays provides enough insight to understand the framework's basic structure.

The Technique Dimension The technique dimension consists of a list of behavioral blocks. A behavioral block is structured partially following Fogg's Behavior Model, which asserts that behavior change is only possible when three factors - motivation, ability, and a trigger - are present at the same time (Fogg, 2009).

Aligning with Fogg's Behavior Model, each block is associated with a design technique and consists of a technique name, a technique summary, an associated trigger for user action and example implementations. The user's end goal for the task they are currently engaged in serves as their motivation. The user's ability has previously been taken into account when the user role was determined within the process dimension and the process was defined and structured accordingly.

Based upon Fogg's persuasive technology strategies, we formulated a set of techniques derived from Wildeboer et al. (2016). The strategies of reduction, tunneling, tailoring, and personalization were retained in their original form as described by Fogg, while the techniques of self-monitoring, simulation, and rehearsal were adapted to better align with the specific requirements of our use case and to enhance usability (Fogg, 2003; Wildeboer et al., 2016).

Our refined set of techniques consists of the following techniques:

- Reduction
- Tunneling
- Tailoring
- Personalization
- Self-Monitoring
- Simulation
- Rehearsal

An exemplary, detailed view of one of the selected techniques incorporated in a behavioral block can be seen in figure 4.2. Further details on the other techniques are available in the appendix chapter A.1 in the demonstration's adapted framework, shown in figure 1 and in the appendix chapter B.

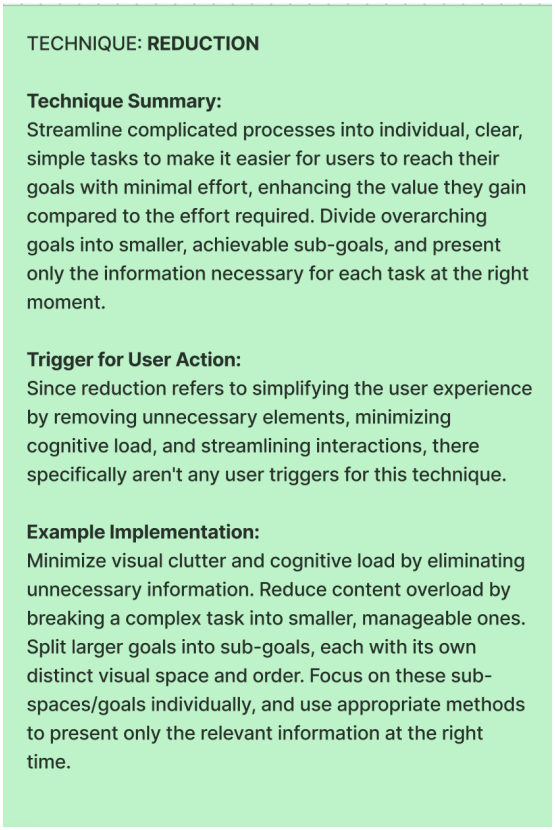


Figure 4.2: The behavioral block of the reduction technique

Behavioral Block Specifics Each block is given a name derived from Fogg's persuasive technology strategies (Fogg, 2003) and Wildeboer et al. (2016), which acts as an identifier not only for the behavioral technique itself but the whole behavioral block. A technique summary provides a quick overview over the key elements that make up and guide the technique, the technique's overarching goals and a road map on how to achieve those on a conceptual level.

Since Fogg's Behavior Model states that even if a user is able and motivated to fulfill a task, it takes a trigger to create momentum within them, a trigger for user action is also provided (Fogg, 2009). Finally, to provide hands on advice, each block contains an example implementation on how the specific technique could be implemented. The aim is for the conceptual information to be tangible and easily accessible for a developer who is not experienced in executing design.

As mentioned before, the technique dimension offers a set of behavioral design techniques to start off with, which are based on Fogg's strategies for persuasive technologies. These techniques offer a sufficient base for a behavioral design optimization toolkit and can - but don't have to be used. We consider them to be a valuable starting point, especially for developers or users of the framework that have no or little design experience or don't want to spend too much time going into the material, but rather use the framework to avoid having to spend the time preparing but instead get a roadmap to easy improvements.

Adding or removing techniques or building a new set from scratch fitted to a specific use case is up to the developer using the framework. This is recommended for power users or users with design expertise only though, as it requires great insight into the behavioral design space. The selection of techniques are crucial make or break building blocks of the whole NUDGE framework. Without appropriate techniques, developer intents can still be mapped to the provided process steps, but the actionable part of taking those intents and building strategies that make them take effect on user behavior will not be able to take place.

The Developer Intents Once the process is well defined and a selection of behavioral blocks has been made, goals need to be formulated for each individual process step in the form of developer intents. A developer intent represents the specific purpose or desired outcome that the developer envisions for the user within a particular process step. These intents should reflect the underlying objectives of the application or process and act as a guiding principle for structuring developers' sub-goals and user interactions at each step. It serves as the rationale for how and why a process step is designed, ensuring that a user is guided toward the intended behavior.

A developer intent can vary across a wide spectrum of sizes, from a small goal, such as drawing a user's attention to a specific UI element — often achievable

by modifying just one element — to a larger objective, like reducing stress and building trust. The latter typically requires adjusting multiple elements and the way they interact with one another. An example we will revisit later in our demonstration chapter 5 is the intent to direct a user's attention toward potential intrinsic or extrinsic rewards gained from contributing, in order to increase their perception of the value of contributing to a project within the open source collaboration space.

Assembling the Framework by Mapping the Dimensions To assemble the framework, the developer intents defined earlier for each process step are mapped to the available behavioral techniques. This mapping creates a structured pathway of actionable steps. By establishing a context-specific "if-then" connection between a (through the developer intent) specified goal and a strategy to reach it, the framework provides clear guidance tailored to each goal or intent, serving both as a roadmap and a practical action plan for achieving objectives.

4.4.2 The Framework's Structure: Why it Works

How it Impacts the Artifact: Why it Works on the Final Product The entire framework and its effects on the final product build over multiple levels. First, the problems and goals must be accurately identified. Next, the appropriate solution strategies for the identified problems must be determined. Finally, these solutions must be applied and be both suitable and effective.

Since the techniques applied in the final step of this framework are based on Fogg's persuasive technology strategies, they have already been validated through his work and subsequent studies (Fogg, 2003; Wildeboer et al., 2016). This ensures that the final step — the effectiveness of the solutions — has already been established and proven. The critical focus for this framework's effectiveness, therefore, lies in identifying the correct solutions, hence the appropriate behavioral design techniques. To achieve this, it is essential to first define the preliminary context and starting conditions accurately and then determine the corresponding goals.

To properly evaluate the preliminary context and starting conditions, we placed particular focus on adapting the framework to a specific use case. This includes requiring the developer to properly define their process to be optimized and to clearly describe the roles and abilities of the users involved. This step serves not only to create clarity but also to integrate Fogg's Behavior Model alongside Fogg's persuasive technology strategies within the framework.

Fogg's Behavior Model, as illustrated in figure 4.3, is a framework designed to support the creation of persuasive technologies by identifying the threshold at which users gain sufficient momentum to take action (Fogg, 2009). It states that successfully indicating a behavior relies on a user's motivation and ability, and can

be influenced by an intervention trigger (Toledo et al., 2018). Hence, for a user to act, a trigger must be present, and the combination of their motivation and ability must exceed a certain threshold. By embedding Fogg's Behavior Model into the NUDGE framework, we ensured that the strategies applied within the artifact are tailored to the specific user group. Therefore, we incorporated all three cornerstones of Fogg's Behavior Model, ability, trigger and motivation, into the NUDGE framework.

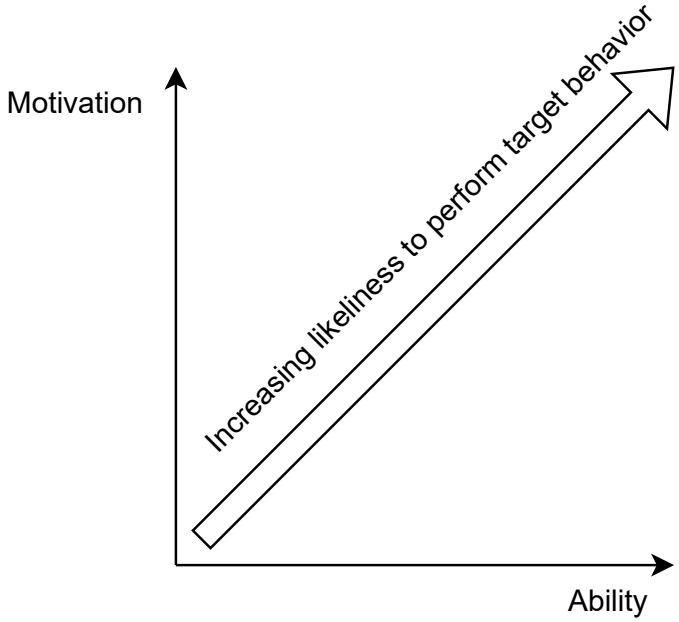


Figure 4.3: Fogg's Behavior Model derived from Fogg, 2009

Since the user's abilities are a cornerstone of Fogg's Behavior Model, we incorporated that part through the definition of user roles and their abilities within the process definition of the framework's adaptation phase, enabling the alignment of strategies with user-specific characteristics later on. Triggers and motivation, on the other hand, are derived from the use case, process, and application context. Examples of triggers have been embedded into the behavioral blocks, while user motivation often stems from the overarching goal of the defined process.

With the preliminary context and starting conditions clearly defined, the next step is to determine the correct goals and identify the appropriate solution path-

way towards those goals. We aim to find those solution pathways through our method of defining developer intents and mapping these to predefined behavioral techniques. By establishing a logical chain of dependencies, as can be seen in figure 4.4, the framework provides a step-by-step action plan, enabling a systematic identification of the most effective solution strategies.

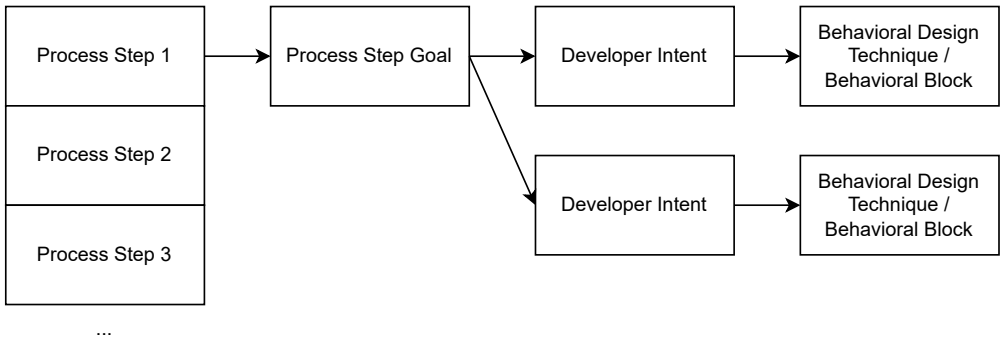


Figure 4.4: Example pathway of mapping a process step to a behavioral technique

Usability of the Framework: Why It Works for Developers and Non-Designers We chose to represent the framework as a three-dimensional visualization encompassing the process, technique, and developer intent dimensions. Eberhard (2021) highlights that information visualization can enhance both the quality and speed of decision-making by alleviating cognitive load and aiding in the understanding of complex information. Furthermore, visualizations can be crucial for cognitive processing and can influence advanced cognitive functions, including problem-solving, sense-making, and decision-making (Parsons and Sedig, 2013). By employing the framework as a kind of visualization (figure 4.1), we aim to simplify decision-making and enhance both the user experience and usability of the framework for developers.

We applied Fogg's Behavior Model not only within the framework's behavioral blocks but also to the framework itself on a meta level. We implemented the motivation, trigger and ability cornerstones of Fogg's Behavior Model into the framework's structure as well to facilitate usability for the developer.

In this context, the developer intent functions as the source of motivation, providing clear objectives. The ability threshold is intentionally kept low, as the framework is designed to be accessible even for individuals without prior knowledge of behavioral design, aligning with its purpose as a tool for non-design professionals. The trigger at this meta level is established through the framework's mapping of developer intents to actionable examples of behavioral techniques.

Through this approach we aim to minimize entry barriers into using the framework and simplify the process of adopting and implementing behavioral design improvements.

Finally, the usability of the framework was iteratively tested and refined through various cycles of feedback and improvement in the evaluation chapter 6. This process allowed us to identify and address potential pain points, ensuring that the final framework is as intuitive as possible and effectively supports its users.

4.5 How to Adapt and Apply the Framework

As mentioned before, the framework is employed in two distinct phases: the adaptation phase and the application phase. The adaptation phase serves as the first and critically important step, during which the framework is tailored to the specific use case. Since any modifications aimed at nudging user behavior are highly context-dependent, it is essential to have a framework that is also context-specific. Therefore, adaptation must always be the initial phase.

Once the framework has been adapted to the use case, the application phase follows. In this phase, the framework's action plan, now adjusted to the specific case and situation, is implemented. Hands-on changes are now applied to the artifact itself, realizing the intended modifications or nudges to user behavior.

4.5.1 Adapting the Framework to a Use Case

The initial step in adapting the NUDGE framework to a specific use case involves defining the process in which the user behavior is intended to be influenced and guided through nudging. To assist developers in clearly outlining and specifying the process and its individual steps, we have developed a set of practical guidelines to follow. These rules of thumb are designed to provide structured support and ensure a comprehensive understanding of the process. They were logically inferred from patterns and insights gained during our hands-on experience working with the framework in the demonstration, which will follow in chapter 5. By reflecting on how we defined the process and its individual steps and observing any pain points we had, we gathered practical recommendations that could help developers effectively define their process to adapt the framework.

Defining and Structuring the Process: Rules of thumb for Defining Process Steps

1. **Clarity and Precision of Titles:** Each process step should have a concise and precise title. This title, when viewed within the context of the overall framework, should be sufficient to understand the content and purpose of the process step without requiring further information.
2. **Brevity of Description:** A process step should be small enough to be described in a single, short sentence.
3. **Clear Focus on a Single Goal:** Each process step should have one overarching goal. If two distinct goals are identified, the step should be split into two separate steps. These steps can then be rearranged as needed to maintain the logical flow of the overall process. Alternatively, consider a higher-level abstraction that can encompass both goals.
4. **Avoiding Goal Overlap:** A process step should be large enough to justify a single, clear goal that is distinct from other process steps. If the goal of a current process step overlaps with that of a previous or subsequent step, these steps should be combined or unified, and the overall process should be rearranged to ensure consistency.
5. **Non-Competing Goals:** The goals of different process steps should not compete with one another. If the goal of a step is unclear or indistinct, reevaluate the necessity of the step. Unnecessary steps can add complexity while offering little to no positive value, which may negatively impact the experience of both the developer using the framework and the end user of the resulting artifact.
6. **Clear Developer Intents:** Developer intents should be easily derivable from the goal of each process step. If a developer intent cannot be clearly identified from the process step and its goal, the step should be reassessed for its relevance.
7. **Parallel Steps and Synergies:** When process steps are set in parallel, avoid creating synergies between them. Each step should be treated individually in terms of goal setting and developer intent deduction to maintain clarity and avoid unnecessary overlap.
8. **Measurable Progress:** There should be a measurable difference in the end user's knowledge or position within the artifact from one process step to the next. This ensures that each step adds clear value to the user's progression throughout the process.

These guidelines are designed to help developers streamline the process, minimize ambiguity, and establish a logical progression of tasks. As the subsequent steps

in the framework's use depend on a well-defined process, ensuring its clarity and structure is crucial. Since this is also the first — and often the most time-consuming — step in the adaptation phase, we aim to provide maximum support to help developers overcome the initial barrier of working with the framework.

Understanding User Roles and Their Relevance in the Process When applying behavioral techniques onto specific processes, it is crucial to identify and analyze the different user roles involved. This includes examining each step of the process and considering which user role is responsible for completing or navigating through these steps. A thorough understanding of user roles helps ensure that the resulting design aligns with the users' needs and abilities.

Based on our experience with defining user roles, key questions to consider may include:

- What are the users' abilities? For example, how technologically skilled are they?
- What prior knowledge do they possess about the artifact or system?
- How experienced are they with the artifact? Are they engaging with it for the first time, or are they recurring users?
- Do they occupy specific roles, such as administrators, with distinct responsibilities?

Understanding user abilities is critical, as ability is one of the foundational elements of Fogg's Behavior Model. It is key to aligning behavioral techniques with the user's capability to engage. Behavioral techniques can only take effect and influence behavior if they are tailored to the users' abilities and are delivered with an appropriate level of support and context.

It is essential to differentiate between the concept of a user role and an individual user. A single user may assume multiple roles within the process, while a specific role may be filled by multiple users throughout a process.

Defining a Set of Techniques The NUDGE framework incorporates a foundational set of seven behavioral techniques, which serve as the primary tools and strategies of the framework's toolkit. Selecting the appropriate techniques is crucial for ensuring that the framework effectively facilitates the desired behavioral changes. Having a tool available for each part of a use case is essential; otherwise, certain aspects may not be properly addressed.

Given the complexity and width of the field of behavioral design, as well as the intricacies of human behavior, we strongly recommend using the established base set wherever it is applicable. Changes to this set, such as adding, modifying, or

removing techniques, should only be undertaken with a solid understanding of the field. Improperly removing a critical technique could undermine the framework's effectiveness in certain areas, potentially compromising its ability to achieve the intended outcomes.

For cases where new techniques are introduced, we advise to maintain the same structural format used in the existing techniques. This involves embedding new techniques within a cohesive behavioral block and integrating triggers as well as example implementations. Preserving this structure ensures that the newly added techniques align with the framework's design principles and maintain compatibility with Fogg's Behavior Model, which is interwoven into the framework's structure.

Formulating Developer Intents: Rules of Thumb and Technique Mapping The final step in adapting the framework to a specific use case is formulating developer intents for each individual process step and mapping these intents to the most suitable behavioral techniques. This stage is particularly critical as it encapsulates the goals, intentions, and desired outcomes of each process step.

Similar to the guidelines we established for defining the process, we have also created guidelines for formulating developer intents for each process step, based on our experience with the framework. Each process step should have at least one developer intent that aligns with its goal. These developer intents represent the goal of the process step, the developer's intentions behind it, and the desired outcome they aim to achieve.

1. **Alignment with Process Step Goals:** Developer intents should be clearly aligned with the goal of each process step.
2. **Non-Conflicting Intents:** Developer intents within a single process step should not conflict or compete with one another. Conflicting intents can create mixed signals, leading to confusion and decreased usability. If the process step is meant to address multiple user groups (e.g., beginners vs. power users), ensure that competing developer intents are either resolved or separated into distinct steps.
3. **Granularity of Intents:** Each developer intent should be singular, not composed of multiple objectives. If an intent contains multiple components, it should be split into smaller, clearly defined intents.
4. **Avoiding Overly Broad Intents:** A developer intent should be specific and measurable. If an intent is too vague or applies to multiple process steps, it may need to be refined, split, or made more precise to fit the framework effectively.

5. **Scalability of Process Steps:** If a process step lacks a developer intent, it may be too small or poorly defined. Consider combining it with the previous or subsequent step to create a more meaningful context. Conversely, if multiple competing developer intents exist within a single process step, the step itself may be too large and should be split into smaller, more focused steps.
6. **Measurable Fulfillment:** It should be clearly defined when and how a developer intent is fulfilled. This includes specifying the action or outcome that achieves the intent. If this is not possible, the intent may be too vague or large and should be refined into smaller, measurable objectives.
7. **Techniques and Intents:** While developer intents are often mapped to specific techniques, not every intent must result in the incorporation of its corresponding technique. The framework should allow flexibility in choosing the most appropriate techniques without mandating their use in every scenario.

Once the developer intents are formulated, they should be mapped to the most suitable behavioral block, thereby linking each individual process step to a behavioral technique based on its intended goal. There is no single "correct" mapping; an intent may align well with multiple behavioral blocks, and the mapping process can be highly subjective. What's important is that developers map their intent to the behavioral block they find most appropriate, as they will be the ones implementing the changes. Ensuring that the mapped pathways align with their way of thinking is crucial for the successful application of the framework within their specific context.

By linking developer intents to behavioral techniques, this step establishes a clear "if-then" action plan that is straightforward to implement. This mapping ensures that each process step is directly connected to an actionable strategy, making the framework practical and effective in its application.

4.5.2 Applying the Adapted Framework to a Specific Use Case

Once the adaptation phase of the NUDGE framework is complete, the result is a comprehensive "if-then" action plan that explicitly links the defined goals within each individual process step of the use case to the available behavioral techniques.

The next phase, the application phase, involves hands-on implementing this action plan within the artifact. This first requires identifying the specific parts of the artifact associated with each process step which are referenced by the developer intents mapped to those steps, and then utilizing the chosen behavioral techniques to address the identified goals. The framework's explanations and

example implementations for each technique serve as a guide for incorporating these elements into the artifact seamlessly. By following this approach, the implementation process becomes straightforward. Developers view a specific process step, identify its relevant parts within the artifact like the associated parts of a product's user interface, refer to the developer intents and mapped behavioral techniques, and implement the suggested strategies to align the artifact with the desired outcomes. The clarity and structure of the action plan ensure that the process is efficient and actionable.

Since the framework provides an action plan that connects behavioral techniques as strategies to goals defined by the developer intents instead of specific design suggestions, it's designed to be reusable. As long as the process itself remains unchanged, no further adaptation of the framework is required. If, however, the process evolves, some aspects of the framework may need to be readapted or updated to align with the new context.

A practical demonstration of an application of the framework can be found in the following demonstration chapter 5, where the NUDGE framework is adapted and applied to the illustrating example of an open source collaboration workflow. The demonstration shows how the framework can be effectively utilized in real-world scenarios to enhance user interaction and engagement.

The IYKYK Principle

To underscore the importance of the thorough adaptation of the NUDGE framework to the specific use case, we recommend applying the "If You Know, You Know" (IYKYK) principle. It emphasizes that for the framework to function optimally, it must be carefully prepared and adapted to the specific use case. User behavior — and the ability to nudge it — is highly context-sensitive, meaning that any attempt to influence or optimize it requires a tool that is equally context-specific. Therefore, it is important to have tailored approaches that align closely with the particular circumstances, users and goals of the use case.

To achieve this context specificity, the framework must be carefully aligned with the exact process it is intended to support. This forms the foundational basis upon which all subsequent actions and outcomes are built. The IYKYK principle tries to showcase this process by providing a clear pathway:

- **If you know your process:** A well-defined, clearly scoped process with distinct and unambiguous process steps provides the essential groundwork. A structured understanding of the process ensures clarity in the subsequent stages.
- **Then you will easily know your developer intents:** From a well-defined process, it becomes straightforward to derive precise developer in-

tents. These intents are tailored to each process step and aligned with the overarching process goals, ensuring they are both actionable and contextually relevant.

- **Then you will easily know the behavioral technique to use:** Clear and specific developer intents lead to an easier selection of appropriate behavioral techniques and strategies. A well-mapped intent provides a direct link to the most suitable technique for achieving the desired behavioral outcome.

This principle highlights the importance of investing time and effort in the preparatory adaptation phase. By ensuring that the process, intents, and techniques are all clearly defined and aligned, the framework can function seamlessly and effectively to achieve its intended behavior-changes.

4.6 Strengths, Limitations, and Opportunities for Extension by Design

The conceptual framework provides insights into how usability challenges can be addressed by integrating behavioral design principles. Through practical guidelines and a flexible, adaptable structure, the framework aims to empower developers to implement user-centered design changes without requiring extensive design expertise. However, as with any tool, it's design also comes with limitations and areas where it could be further refined to increase its applicability and impact. In the following sections we will reflect on the framework's design.

4.6.1 Strengths

Adaptable, Modular, and Customizable Design One of the key strengths of the framework is its adaptability. Developers can tailor process steps, techniques, and intents to suit specific use cases or scenarios, ensuring its relevance across diverse applications. Its three-dimensional structure — comprising process steps, techniques, and developer intents — provides a systematic and flexible approach to addressing usability challenges, making it a robust tool for various contexts.

Additionally, considering its modularity, the mapping and therefore the size of the adapted framework can be either super simple and small or big and complex depending on the applied use case and its complexity.

Integration of Behavioral Design Principles The framework leverages Fogg's Behavior Model and Fogg's persuasive technology strategies to provide a theor-

etically grounded approach to influencing user behavior. This alignment with behavioral design principles reinforces the framework's credibility and utility.

Practical and Actionable Designed with practicality in mind, the framework offers developers a pre-defined toolbox of behavioral design techniques, along with actionable insights, in the form of behavioral blocks. By encouraging developers to define clear processes with specific goals, it ensures that developer intents are clearly defined, measurable, and actionable, increasing the likelihood of achieving desired outcomes. This makes it accessible to developers without formal training in design or usability, reducing cognitive load and providing structured guidance. By simplifying the integration of behavioral design techniques, the framework empowers developers to make impactful design changes without requiring extensive expertise.

Open Source Relevance The framework was structured with open source environments in mind, where usability can take a backseat to functionality. By aiming to be most suitable for addressing usability challenges unique to developer-driven open source workflows, the framework aims to promote inclusivity and lower barriers to collaboration.

Iterative and Reusable The framework is able to support iterative refinement. Developers can repeatedly apply and adapt the framework to refine usability of their artifact over time, ensuring that it remains effective and relevant as workflows evolve. This reusability can make the framework a long-term solution for ongoing improvements.

4.6.2 Limitations

Reliance on Accurate Process and Intent Definition The framework's effectiveness depends heavily on well-defined process steps and developer intents. Ambiguities or inaccuracies in these elements can lead to suboptimal results, limiting the framework's effectiveness.

Initial Learning Curve and Investment Despite being designed for accessibility, the framework may present a steep learning curve for developers unfamiliar with behavioral design or usability concepts. Additionally, the initial effort required to correctly adapt the framework may deter its primary target audience seeking quick and easy-to-apply solutions.

Dependence on Pre-Defined Techniques While the framework includes a curated set of behavioral design techniques, it may not comprehensively address all use cases or advanced scenarios. Expanding or customizing these techniques

requires expertise, potentially limiting the framework's usability for less experienced developers or those working in niche contexts.

Complexity in Large-Scale Applications In large-scale systems with numerous parallel processes and diverse user roles, mapping developer intents and techniques can become complex and unwieldy. Conflicts may arise from overlapping or competing intents, particularly in scenarios involving multiple user groups.

Focus on Isolated Processes The framework is designed to focus exclusively on a single, isolated process without considering external or intersecting processes. While this ensures precision and clarity within the targeted process, it does not account for interactions or dependencies with other processes that may overlap or cross paths.

Assumed Generalization of the Framework Currently, the generalization of the framework from the specific use case of the open source collaboration workflow remains untested. While we assumed that the framework could be applied effectively to other domains and use cases, this has not yet been empirically validated.

4.6.3 Opportunities for Extension

Integration of Cross-Process Dependencies As for now, the framework only takes a single isolated process into consideration. It could be extended to account for intersecting or overlapping processes, enabling developers to optimize workflows that involve multiple interconnected processes. This could for example include tools for mapping and visualizing dependencies and synergies between processes, offering a more holistic approach to applying behavioral design across complex systems.

Enhanced Guidance and Examples Providing detailed templates, case studies, and examples could help developers define process steps, developer intents, and techniques more easily and effectively. Additional resources for handling complex workflows and diverse user roles could promote the framework's accessibility and usability for a broader audience.

Technique Expansion The library of behavioral design techniques could be expanded to include advanced or industry-specific strategies. Pathways for developers to intuitively integrate custom techniques into the framework could also make it more versatile and adaptable to unique use cases.

Testing Generalization and Broader Applicability It would be valuable to test whether the generalization from the specific use case of the open source collaboration workflow was successful. Through further evaluation we could determine whether the framework could also be applied to different areas and use cases beyond the one initially explored.

Automation and Tool Support The development of software tools or plugins could automate parts of the framework, such as mapping intents to techniques or generating process flow diagrams. AI-driven recommendations based on developer inputs and use case characteristics could further streamline the application of the framework, making it more user-friendly and efficient.

4.7 Assessment of Met Design Requirements

The NUDGE framework's design was developed to fulfill the design requirements stated in chapter 4.1. The following table evaluates how well these requirements were met by the framework's design.

Functional Requirements	**Met**
Problem Definition	Yes
Solution Mapping	Yes
Implementation Support	Yes

Table 4.1: Assessment of met functional requirements

Non-Functional Requirements	**Met**
Scalability	Partially
Flexibility	Yes
Simplicity and Accessibility	Partially
Behavioral Design Alignment	Yes
Inclusivity in Open Source Contexts	Yes
Reusability	Yes (assumption)
Transferability	Partially (untested)

Table 4.2: Assessment of met non-functional requirements

Functional Requirements

- **Problem Definition:** The framework provides a structured methodology for developers to define the usability challenges within their processes. The adaptation phase is designed to guide developers in defining process steps and user roles, aligning with this requirement.

- **Solution Mapping:** The mapping of the process dimension to the technique dimension through developer intents effectively connects specific usability problems to actionable strategies, satisfying the requirement for solution mapping.
- **Implementation Support:** By providing practical steps, triggers, and examples for implementing behavioral techniques, the framework's design meets the need for implementation support. Developers are given clear, actionable pathways to incorporate behavioral design. Additionally, the rules of thumb for both the process definition and developer intent formulation support the developer in their process.

Non-Functional Requirements

- **Scalability:** The modular structure of the framework's design, adaptable to processes of varying complexities, meets the scalability criterion only to an extent. Considering the freedom provided by the framework, the mapping itself can range from a small, simple size to a big and complex one depending on the applied use case. We are assuming, however, that since cross-process dependencies and synergies aren't considered yet, the framework might not be applicable too well for large-scale projects. Therefore, the scalability requirement is met only partially.
- **Flexibility:** Since the framework is designed to function without needing developers to have any prior expertise, it accommodates users with varying levels of expertise, offering detailed guidance for beginners and flexibility for experienced developers. Additionally, the modularity of the framework allows it to grow with the size of the use case at hand. Thereby fulfilling the flexibility requirement not only for its user base but also the framework itself.
- **Simplicity and Accessibility:** While the framework's design achieves simplicity in structure, an initial learning curve exists, particularly for developers unfamiliar with behavioral design. Future iterations could address this gap with enhanced documentation and tooling support.
- **Behavioral Design Alignment:** The integration of Fogg's Behavior Model and persuasive technology strategies ensures the framework's alignment with behavioral design principles. This consistency bolsters the framework's credibility.
- **Inclusivity in Open Source Contexts:** The framework is built to address open source challenges by prioritizing usability and user behavior. This can be a valuable way to reduce barriers to contribution, meeting this specific non-functional requirement.

- **Reusability:** Based on the framework's structure and intended use, we assume that it is feasible to adapt and reuse it across multiple projects. It would thereby fulfill this requirement, although this has not been empirically evaluated yet.
- **Transferability:** The framework's applicability to domains beyond the open source collaboration space remains untested, leaving this requirement partially unmet. While the only precondition for the framework to be applicable to a use case is, that it must be possible to break it down into a well defined process, empirical validation is necessary to confirm the framework's generalizability.

4.8 Summary

The Solution Design chapter introduced the NUDGE framework, a structured tool for integrating behavioral design principles. The framework's three-dimensional structure, encompassing process steps, developer intents, and behavioral techniques, was elaborated upon, alongside its theoretical foundation in Fogg's Behavior Model and persuasive technology strategies.

Key points include:

- **Conceptual Framework:** The adaptation and application phases ensure that the framework is tailored to specific use cases while remaining adaptable to evolving contexts.
- P**rocess Customization and Developer Intent Mapping:** The emphasis on defining a clear, measurable process and formulating goals and intentions for each step through developer intents enhances the framework's effectiveness.
- **Behavioral Blocks:** A curated set of techniques based on Fogg's persuasive technology strategies provides practical action plans for addressing user behavior.

Our assessment suggests that the framework met most functional and non-functional requirements. However, areas for improvement include simplifying the adaptation phase and validating the framework's generalizability across domains. These limitations highlight opportunities for further research and refinement.

In conclusion, the NUDGE framework provides developers with an accessible tool to influence user behavior.

5 Demonstration

For this demonstration, we focused on the open source collaboration workflow, specifically using the JValue Hub [1], a platform for open data, as the illustrative example. As mentioned in the problem identification in chapter 2, open source environments often lack prioritization of user-centered design, which can lead users to favor the more intuitive interfaces of proprietary software (Raza and Capretz, 2015).

As described in chapter 3, the primary objective of this thesis is to address this by incorporating behavioral design principles into the development process, aiming to make tools more intuitive and user-friendly. Following the terminology of chapter 4, we will use the term "user," as the end-user of the application whose behavior we seek to influence. On the other hand, "developer" refers to the person using the framework to optimize the product. In the context of this demonstration, we, the authors, act as the developers.

This demonstration aims to show how developers can use the NUDGE framework to identify areas for improvement, select suitable behavioral techniques, and implement them within the context of an open data platform like the JValue Hub. With this approach we aim to improve collaboration by making the platform more intuitive and reduce barriers for both new and experienced users.

5.1 Selecting Process Steps to Demonstrate

We created a sampling matrix of all process steps available within the open source collaboration process and the techniques we mapped to a developer intent, which can be seen in figure 5.1. We then used polar sampling to select steps that represent a variety of different distributions of applicable techniques. We opted for polar sampling as it enabled us to select the most differently characterized steps covering the widest range of the framework's different application scenarios. The selection of steps was split into two groups, small steps and complex, multitask

[1] https://hub.jvalue.com/

steps. Since we used the open source collaboration process for both the demonstration and evaluation of the framework, we made sure that both of them were provided with steps from the small group as well as steps from the complex group. This approach ensured that the broadest possible spectrum of different applications of the framework and various characteristics of the process steps could be tested and covered.

process step / technique	reduction	tunneling	tailoring	personalization	self-monitoring	simulation	rehearsal
initiate choice for forking		X		X	X	X	
fork project	X	X	X		X	X	
alter forked project							
initiate choice for contribution		X	X	X		X	
pre-steps for contribution request	X	X	X		X	X	X
create contribution request	X	X	X		X	X	X
interact on contribution request changes	X	X	X		X	X	X
initiate choice for contribution integration		X				X	
pre-steps for contribution integration	X	X			X	X	X
integrate/reject contribution request	X	X	X		X	X	X

small step
complex, multi task step
not applicable
in demonstration
in evaluation

Figure 5.1: Sampling matrix for selecting process steps to demonstrate and evaluate

5.2 Adapting the Framework to Our Open Source Collaboration Use Case

Following the approach outlined in the solution design, we began by defining the overall process and breaking it down into individual process steps, guided by the process definition principles outlined in chapter 4.5.1. Alongside this, we established distinct user roles and associated each process step with the relevant role.

Once the process structure and roles were in place, we identified a set of behavioral techniques to apply. For each process step, we then formulated developer intents in line with the intent definition guidelines also provided in chapter 4.5.1. Finally, we mapped these developer intents to the selected behavioral techniques, creating a cohesive framework that's well adapted to our use case.

Defining the Process and User Roles To understand and abstract the open source collaboration process, we analyzed various workflows and their steps to

identify general commonalities and differences. For this purpose, we examined three widely used and well adapted collaboration tools: the code collaboration platforms GitHub [2] and GitLab [3], and the project management tool Jira [4].

In our analysis, we identified two primary user roles involved in the collaboration process:

1. **Artifact User:** This role typically represents a user who interacts with the project by making contributions. While they are often new or less experienced, they could also be a highly skilled power user depending on their familiarity with the project and tools. Their permissions are usually restricted to actions like forking, altering their copy of the project, and submitting contribution requests.
2. **Artifact Owner or Admin:** This role represents a more experienced user with greater knowledge of the project and more extensive permissions. The artifact owner or admin oversees the project, evaluates contributions, and decides whether to integrate changes.

It's important to note that while the two roles can sometimes be embodied by the same person, they remain distinct in terms of responsibilities and permissions. A visual representation of these process steps can be seen in figure 5.2.

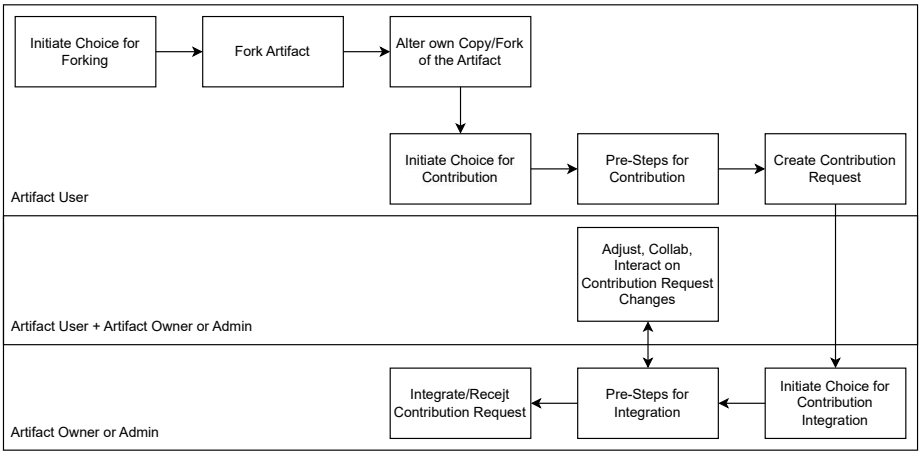


Figure 5.2: The abstracted open source collaboration process

The following are the key steps in the open source collaboration workflow we deducted. Steps 1-6 involve the artifact user, steps 7, 8 and 10 the artifact owner, and step 9 connects them both through interaction.

[2]https://github.com/
[3]https://about.gitlab.com/
[4]https://atlassian.com/de/software/jira

Step 1: Initiate Choice for Forking
[Role: Artifact User]
The code collaboration systems we analyzed all include a mechanism to trigger the user's decision to obtain their own copy of a project, commonly referred to as "forking." This is a critical first step in the open source collaboration workflow, as it enables all subsequent actions leading to contributions.

Step 2: Fork the Project
[Role: Artifact User]
Once the user decides to contribute, they fork the project they intend to alter or contribute to.

Step 3: Make Alterations
[Role: Artifact User]
The forked project, now copied to the user's workspace, is modified according to their intentions.

Step 4: Initiate Choice for Contribution
[Role: Artifact User]
After completing their alterations, the user's motivation to contribute their changes must be prompted. This step initiates the decision to share their work with the original project.

Step 5: Necessary Pre-Steps for Contribution (Optional)
[Role: Artifact User]
Depending on the process, certain preparatory steps might be required before creating a contribution or merge request. These are quite use case specific and could be of technical nature, e.g., including specific automated tests, they could also be of administrative nature, e.g., signing a contributor license agreement.

Step 6: Create and Submit Contribution Request
[Role: Artifact User]
The user formally creates and sends a contribution, also known as a merge request, to the original project.

Step 7: Initiate Choice for Contribution Integration
[Role: Artifact Owner or Admin]
The artifact owner or admin reviews the contribution request and decides whether to proceed with integrating it into the original project.

Step 8: Necessary Pre-Steps for Integration (Optional)
[Role: Artifact Owner or Admin]
Before integrating the contribution, the artifact owner may need to perform technical or administrative tasks, such as testing the proposed changes, reviewing the code for compliance, or ensuring it does not conflict with other project elements.

Step 9: Adjust, Collaborate, Interact on Contribution Requests (Optional)
[Role: Artifact Owner or Admin; Artifact User]
This step involves interaction between the artifact owner and the contributor. They may engage in discussions or iterative refinements to adjust the proposed contribution. The aim is to ensure the changes align with the project's goals and benefit all stakeholders.

Step 10: Integrate or Reject Contribution Request
[Role: Artifact Owner or Admin]
The process concludes with the artifact owner's decision to either integrate or reject the contribution request.

This logical breakdown highlights the essential steps and roles in the open source collaboration process, providing a foundation for the adaptation of the NUDGE framework to the open source collaboration use case.

The workflow demonstrates a well-structured process that closely adheres to the rules for process step design:

- **Clarity and Brevity:** Each step has a clear, concise name and is small enough to be described in one sentence.
- **Singularity of Goals:** Every step focuses on one overarching goal, with no competing intents or goals within or between steps.
- **Logical Flow:** The steps build on one another logically and have a smooth progression with measurable outcomes.
- **Parallel Step Independence:** Optional steps are treated independently to avoid unintended synergies.

We focused on ensuring that each process step in the open source collaboration workflow adhered to the guidelines for process definition we formulated in chapter 4.5.1. Each step was given a clear and concise title, such as "initiate choice for forking" or "fork the project", which accurately reflects the core activity within the step. This ensures that developers can easily understand the purpose of each step without needing additional context. We also made sure that each step could be described in a single, straightforward sentence, such as "the user creates and submits a contribution request".

We focused on maintaining a singular overarching goal for each step. For example, the goal of step 2 is solely to create a fork, and step 3 is about altering the forked project. There is no overlap between steps, ensuring that each step has a distinct and meaningful objective. This structure avoids any competing goals within individual steps and maintains a clear and logical flow from one step to the next.

In cases where there are optional or parallel steps, like steps 5, 8 and 9, we ensured these steps were independent from each other, avoiding any synergies or unnecessary dependencies. This approach aligns with the rule of thumb that parallel steps should not interfere with one another, treating them as standalone activities that can be pursued independently.

We also paid attention to ensuring measurable outcomes between steps. For instance, after step 2, the user has successfully forked the project, and after step 6, a contribution request has been sent. These measurable results help users understand the impact of their actions and their position in the workflow. By following these guidelines, we created a well-structured process that supports effective collaboration between artifact users and artifact owners.

Defining the Technique Space We utilized the fundamental set of behavioral techniques derived from Fogg's persuasive technology strategies as the foundation for our framework. We chose this base set of techniques intentionally because the demonstration serves as a critical foundation for the subsequent evaluation of the framework. By using these core techniques, we ensure that the framework is tested in its most basic and unaltered form, providing a solid basis for analysis.

Defining developer intents After mapping our previously abstracted open source collaboration process onto the framework, we proceeded to formulate our developer intents. We followed a sequential approach, going through each process step and individually defining the intents for each, while adhering to the rules for formulating developer intents as outlined in chapter 4.5.1.

For every process step, we defined the desired outcome: what we, as developers, aim to achieve at that stage. This involved specifying what parameters should change, where the user should be, what information they should have, and whether they should be motivated to take actions they hadn't considered before. We also identified the specific user actions that would lead to those goals, breaking down what each action involves and how the user would need to engage. Further, we considered how users would recognize what they need to do, what information or resources are required, and how these elements are currently provided.

Based on these considerations, we defined user intents for each step. Once the intents were established, we mapped them to the corresponding techniques of the framework. That way we were able to ensure that each step was supported by clear goals and appropriate behavioral techniques.

Some developer intents were repeated and reused because they applied to multiple process steps, while others required only slight adjustments or corrections to fit specific steps more accurately. Each user intent, along with its corresponding

process steps and goals, is visually represented in the appendix chapter A.1 in figure 1.

5.3 Applying the Adapted Framework on the JValue Hub

The JValue Hub is an innovative collaboration platform for support open-data pipelines. It allows users to create and execute data pipelines in a typical open source development style. Additionally, the JValue Hub features an integrated search function for open data.

Within this demonstration we applied the in the previous chapter adapted framework to the JValue Hub's collaboration process. Since the tool itself is partially still in a prototype state and a clearly defined collaboration process still under active development, we used the previously shown generalized and abstracted open collaboration process and we therefore were able to use the previously adapted framework without the need to alter it any further.

5.3.1 Selected Process Steps

For the demonstration, we selected two steps from the open source collaboration workflow, both carried out by the artifact owner or admin:

1. Initiate Choice for Contribution Integration
2. Interact on Contribution Request Changes

Through polar sampling, these two consecutive steps were chosen because they vary significantly in complexity and user engagement. This allows us to showcase a broader spectrum of the framework's application, as adapted for the open source collaboration workflow use case.

The first step, *Initiate Choice for Contribution Integration*, focuses on triggering the artifact owner to review a contribution request. It ensures that the owner takes the initial step to engage with the contribution, setting the stage for deeper involvement.

The second step, *Interact on Contribution Request Changes*, emphasizes community building and dynamic collaboration. This step fosters active exchanges and cooperation between a contributor and the artifact owner or admin.

In the following subsections, the relevant developer intents of those two process steps are outlined, and their implementation through their mapped behavioral techniques is visualized using screenshots of the JValue Hub's user interface.

For better understanding, we grouped these implementations via their associated behavioral technique.

Process Step 1: Initiate Choice for Contribution Integration After the contributor has forked the project, made their changes, and decided to share those alterations with the original project, they submit a formal contribution request. This serves as a proposal where the contributor outlines the modifications they've made. The contribution request is sent to the artifact owner or admin for review, initiating the next step in the collaboration process.

The step *Initiate Choice for Contribution Integration* refers to the moment when the artifact owner or admin is prompted to decide whether they want to review the submitted contribution. This step does not involve the detailed evaluation or integration of the contribution itself, but rather signals the start of the decision-making process. At this point, the artifact owner is prompted to engage with the contribution request and decide whether they will proceed with a deeper review. It is an initial step that sets the stage for the potential evaluation and integration of the changes into the project.

We used the developer intents set through the adapted framework to identify fitting behavioral techniques that nudge the artifact owner toward making the choice to engage with the contribution request. The following techniques and alterations of the JValue Hub's user interface represent the solutions we identified through this process step, designed to effectively encourage the artifact owner to engage with the contribution request and initiate the decision-making process.

For this process step, we concentrated on the project page, as it represents the most probable location for users to be situated when arriving at this specific step. Figure 5.3 depicts the project page of an example project within the JValue Hub, with the subpage for incoming merge requests currently displayed.

Figure 5.4 additionally shows the project page's subpage for a specific pipeline that was run and displays the pipeline's statistics based on the prior run(s).

Tunneling The following changes were made based on the developer's intent to guide the user's attention towards possible intrinsic or extrinsic rewards gained through contribution integration. This could, extrinsically, involve showcasing their position within the community and/or expanding their user base. Intrinsically, it could entail fostering a community, receiving support, or gaining appreciation for their artifact. This intent aligns with and is mapped to the tunneling behavioral block.

To achieve the overarching goal of encouraging users to make the choice to integrate a contribution request, they must first be triggered to notice it. Consequently, the initial goal involves guiding the user's attention toward the ex-

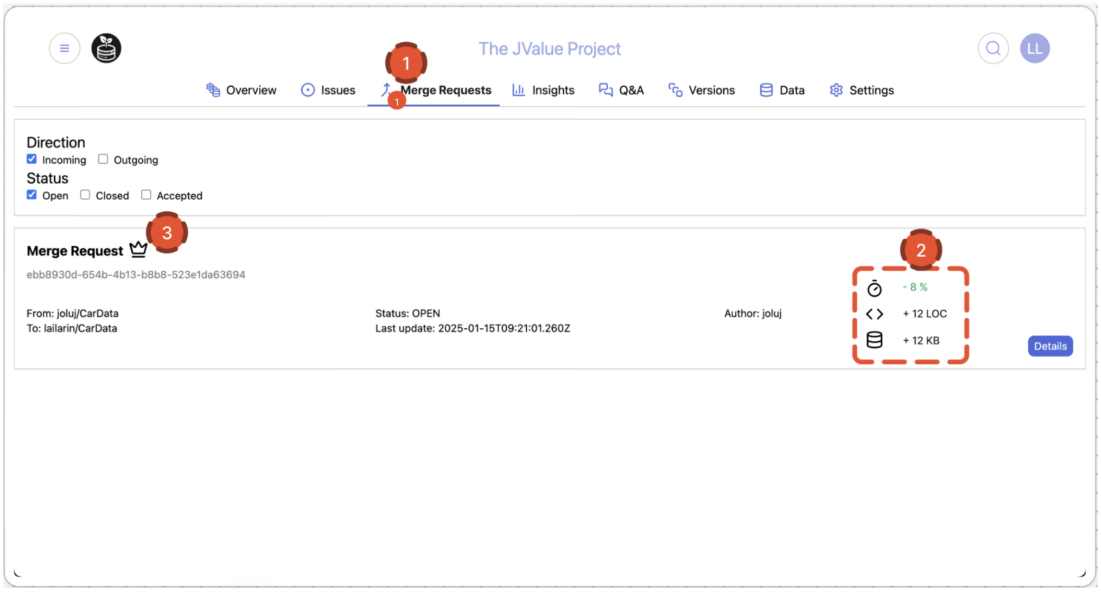


Figure 5.3: Project page: MR subpage

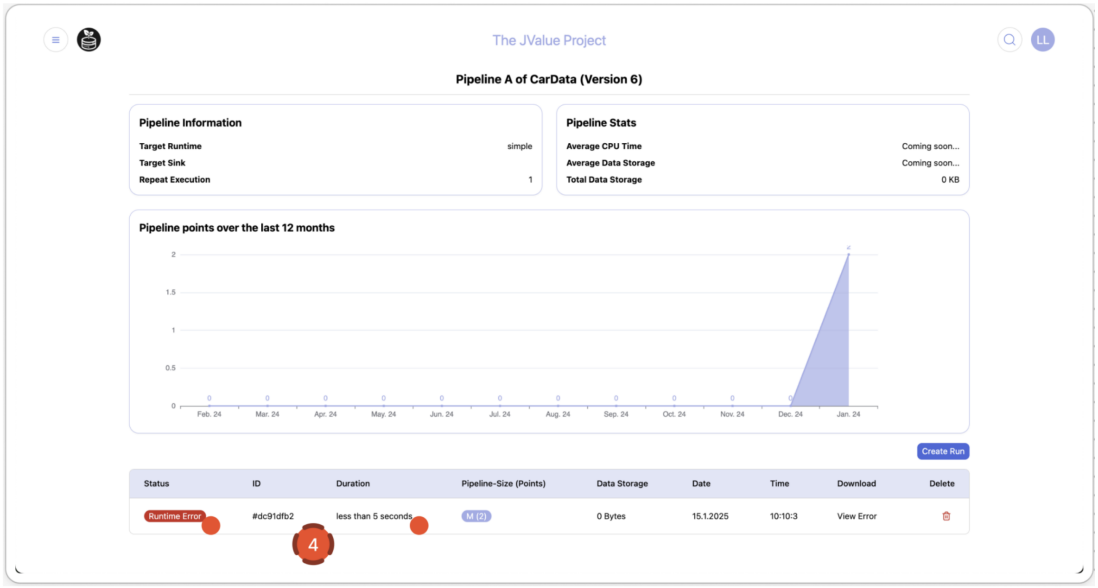


Figure 5.4: Pilepine statistics page

istence of the contribution request, in the JValue Hub referred to as a merge request.

1. **Visual Emphasis in Navigation (Figure 5.3, Change 1)** The first application of the tunneling technique involves visually highlighting an incoming merge request in the tab navigation bar. This is achieved by increasing its prominence within the visual hierarchy: the font of the merge request tab is set to bold, and a visual marker — a circle displaying the number of new, incoming, and unviewed merge requests — is added. These enhancements can be seen in figure 5.3, change 1.

   Since this navigation bar is present across all subpages of a project page, the project owner is consistently notified of new merge requests. This approach draws inspiration from widely-used applications such as Gmail [5], where bold tabs and small circular indicators are associated with new notifications, leveraging existing user biases.

2. **Highlighting Potential Benefits (Figure 5.3, Change 2)** To further motivate users to integrate contribution requests, indicators for changes in the project's metrics were added through three icons and according numbers as seen in figure 5.3, change 2. Icons and prominent color coding — such as green, which is commonly associated with positive outcomes — are employed to enhance the visual hierarchy and draw attention to possible benefits. These modifications are designed to guide the user's focus toward the value-added aspects of the contribution.

3. **Contextual Indicators (Figure 5.4, Change 4)** In figure 5.4, change 4, we introduced indicators similar to those in the merge request tab of the navigation bar. These indicators appear next to a metric when a contribution request is submitted that could potentially optimize that specific metric of the project. It is important to carefully consider the placement of these indicators to avoid conflicting with other user flows on the page. While this approach provides an opportunity to highlight potential fixes directly within the context of the project where issues are identified, its integration should be done with caution to ensure it does not disrupt or overwhelm other critical interactions.

Tailoring The developer intent mapped to the tailoring technique is to reduce the mental blocker of starting an unknown, complex process by minimizing information overload and helping users focus on the task without distractions.

Tailoring refers to adjusting content, layout, and design to meet the individual needs and preferences of users. In this case, users are informed if the account

[5] https://mail.google.com/

submitting a contribution request has previously submitted requests that were successfully integrated into the project.

To achieve this, users with a history of successful contributions are marked with a crown icon, as shown in figure 5.3, change 3. This visual cue signals the credibility and reliability of the contributor, implying that the current contribution request is likely to be valuable. By highlighting prior successful collaborations, this feature aims to reduce uncertainty and foster trust in the contribution process.

Simulation The developer intent for the simulation technique is to reduce the mental load of unforeseeable changes in an unknown system. Additionally, it's to provide a sense of value to what integrating a contribution would mean.

The indicators of changes in the project's metrics as seen in figure 5.3, change 2, also serve to simulate the potential outcomes of integrating the contribution request. By showcasing possible changes to the project's metrics — partially represented in relative terms, such as an 8% decrease in runtime — users can better anticipate and evaluate the possible consequences of their decision.

Furthermore, color coding again enhances the perceived value of these outcomes. The use of green font reinforces the association with positive results, further encouraging user engagement.

Process Step 2: Interact on Contribution Request Changes This process step involves the project or artifact owner actively engaging with the contribution request and the changes proposed within it. Unlike the previous step, which simply aimed to trigger consideration and decision for integration, this step encompasses a more extensive workflow. The primary objective is to guide the user through the multi-step process of reviewing the contributed changes, interacting with the contributor, and evaluating the suitability of the proposed modifications. This includes tasks such as writing comments, discussing the changes with the contributor, thoroughly examining each modification, and determining whether the proposed adjustments align with the project's goals and should be incorporated.

Figures 5.5 and 5.6 both show the overview page of a submitted incoming merge request. Figure 5.5 represents the current state of the page on the JValue Hub, while figure 5.6 illustrates the modified version we altered in accordance with the principles of the NUDGE framework.

Reduction The developer intent for this step emphasizes that when the process or interaction cycle is complex, it is best to break it down into smaller, more manageable steps. Simplifying these steps can enhance the user experience and reduce perceived stress.

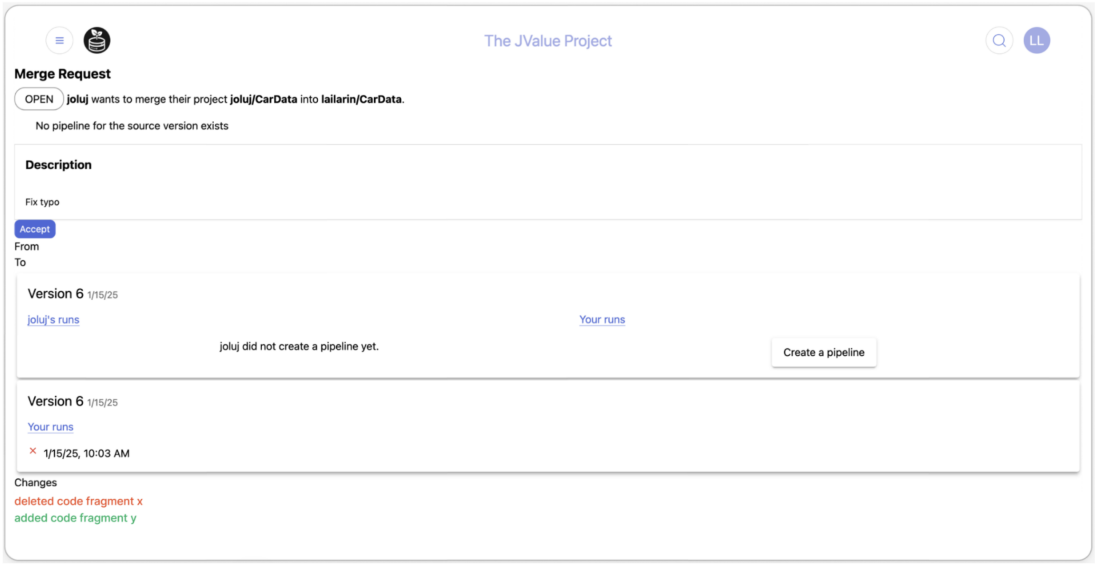


Figure 5.5: MR overview page

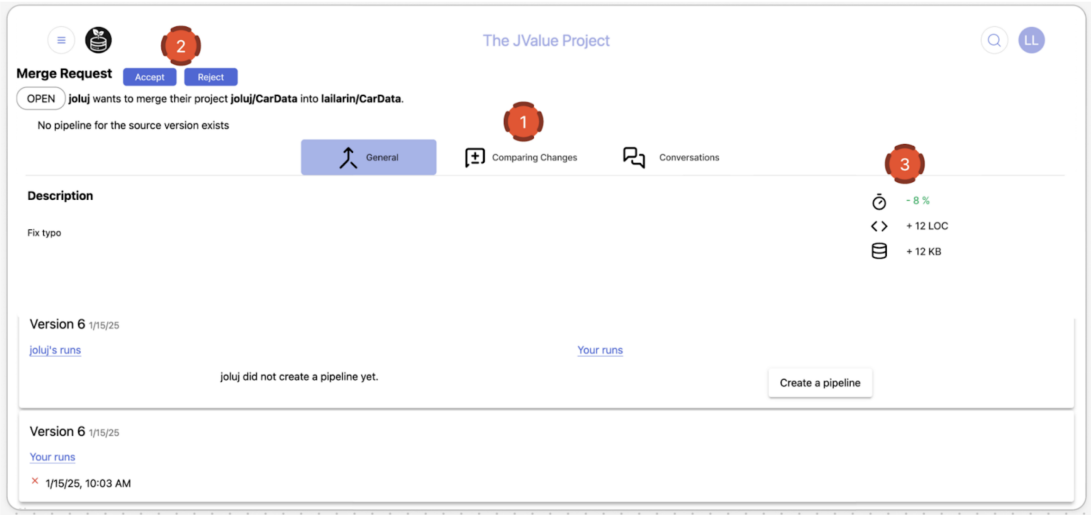


Figure 5.6: Modified MR overview page

Given that this process step involves a multifaceted workflow requiring substantial user input, we aim to simplify it to encourage engagement and reduce information overload and confusion. Furthermore, we want to increase user interaction by providing multiple opportunities and methods for engagement across different parts of the process.

To address the potential for information overload, we divided the displayed information into subpages dedicated to specific categories, as illustrated in figure 5.6, change 1. This approach ensures that users receive only the right and relevant information at the right time, making it easier to process. The categories are organized into navigation bar tabs, similar to those already used throughout the application, creating a familiar interface. A clear visual hierarchy highlights the currently selected category, giving users fast contextual awareness of the displayed content.

For example, as shown in figure 5.6 the initial page provides general context for the merge request, such as its description and versioning details. This contextual information helps users understand where the merge request's changes fit within the project and whether it is relevant to the current version, offering valuable points of reference.

Tunneling The developer intent for tunneling in this specific step is to guide the user's attention to the specific requests made by the other party. This approach aims to streamline the interaction for the user and reduce the complexity of locating relevant changes or requests in a merge request within a single interaction cycle.

To specifically bring the user's attention towards requests made by the contributor, we incorporated a category called "conversations" specifically dedicated to any interactions between the two parties. These, as can be seen in figure 5.7, change 4, provide the contributor with the option to comment on specific changes they've made as well as the project admin with the option to answer. The current user can comment back on changes made and any questions that might arise with it.

In line with this intent, as seen in Figure 5.6, change 2, we repositioned the "Accept" button to a more general location, ensuring it remains visible and easily accessible across all subpages of the navigation bar. This adjustment allows the user to progress to the next step in the overarching workflow without any barriers. Additionally, we included a "Reject" option to provide users with clear choices, enabling smoother navigation throughout the process.

Simulation and Tunneling When analyzing this process step within the JValue Hub, we came to the conclusion that a notable source of user adversity

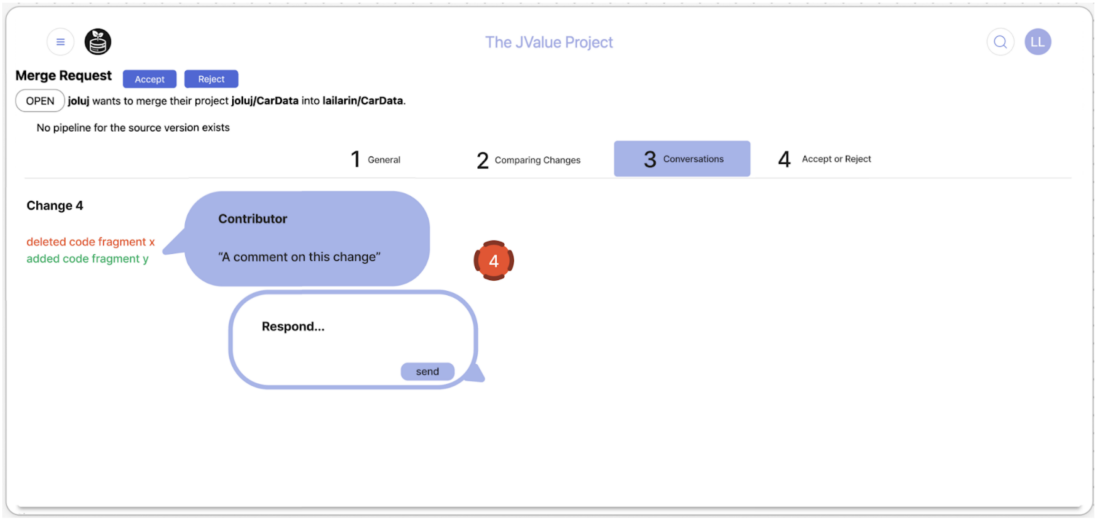


Figure 5.7: Conversations subpage of a MR

here might stem from information overload and the resulting sense of confusion or loss of control. To mitigate this, like shown in figure 5.6, change 3, we again added indicators of changes in the project's metrics, this time to motivate the user to explore the changes in more detail. By showcasing potential benefits, we again want to encourage the user to engage with the proposed modifications and interact with the contributor who submitted the merge request. This approach not only seeks to reduce feelings of overwhelm but also to foster meaningful engagement, aligning with the goals of both simulation and tunneling principles.

Alternative Implementations for Step 2 Figure 5.8 presents an alternative option for dividing the complex overview page into distinct categorical subpages using a navigation tab bar. In this approach, instead of employing icons for categorization, we implemented a numbered list to represent the different categories. This design choice provides an additional layer of structure, making it particularly suitable for processes that involve straightforward progression rather than prolonged engagement in a single step or iterative interactions with the contributor.

Self-Monitoring The developer intent for the technique of Self-Monitoring of this process step is to provide users with a sense of control and an overview of their position within the process. For lengthy or complex workflows, self-monitoring becomes essential to reinforce the user's perception of control. As illustrated in figure 5.8, change 5, one potential enhancement involves displaying progress indicators, like numbered tabs, to ensure the user does not overlook any requests from the contributor. This approach is particularly effective if the process does not heavily depend on iterative back-and-forth interactions. Additionally, includ-

Figure 5.8: Alternative modifications for the MR overview page

ing navigation prompts, such as a "Next" button or progress indicators at the bottom of the page, could further guide users through the process. Finally, as shown in figure 5.8, change 6, positioning the "Accept" and "Reject" buttons at the bottom of the last sub-page of the numbered process also provides a clear and accessible method for concluding the workflow.

Rehearsal In terms of the developer intent for and goal of the technique rehearsal, it's to offer a brief rehearsal of the changes made that can help the user to feel prepared before proceeding with the actual integration.

In the context of figure 5.8, change 8, an optional feature could allow the project owner to see which changes they have viewed already and select specific changes from the merge request that they wish to incorporate. This would act as a rehearsal, enabling the user to preview and evaluate changes before final integration.

Tailoring For tailoring and its intent to provide information and context of the specific step to the user accordingly, the JValue Hub could implement context-specific fields that offer descriptions and guidance for new users. These fields could appear during the first interaction with a merge request or be dismissible, allowing users to opt out of seeing them in the future. However, we decided to not implement this feature in the current use case. Adding such context fields, we concluded, would introduce unnecessary visual complexity to an already detailed user interface. Additionally, the contribution review process is currently still in a prototype state, and as for now misses some features to come, like automated tests or advanced configuration options often seen in tools like GitHub. Since we

would be adding tooltips for unfinished tools, we postponed these elements for a future session.

5.4 Summary

For the demonstration of the NUDGE framework on the open data platform JValue Hub, we used polar sampling to select two steps to apply the framework on: *Initiate Choice for Contribution Integration* and *Interact on Contribution Request Changes*. These steps were chosen to demonstrate a range of complexity within the steps and necessary user engagement.

We adapted the framework's process dimension by analyzing collaboration workflows in tools like GitHub and GitLab and abstracting a more general open source collaboration process. We defined roles for the Artifact User (contributor) and Artifact Owner/Admin (reviewer) and broke down the process into steps and defined our developer intents, both according to the rules of thumb provided in chapter 4.5.1. Each step was then mapped to behavioral techniques through its developer intents.

For the application of the framework on the JValue Hub, we implemented techniques such as:

- **Tunneling:** Highlighted new merge requests with visual cues to prompt action.
- **Tailoring:** Reduced uncertainty by marking reliable contributors.
- **Simulation:** Displayed potential impacts of contributions to simulate possible outcomes.
- **Reduction:** Organized information into manageable subpages to simplify workflows.
- **Self-Monitoring:** Added progress indicators for clearer navigation.

We showcased our implementations visually through modified screenshots of the JValue Hub's user interface guided by more detailed explanations of our alterations and reasonings.

6 Evaluation

This chapter focuses on the evaluation of the framework. It starts by defining the success criteria of our objective definition in chapter 3 in more detail. We then detail the evaluation method and process, followed by a discussion of the results. Finally, we examine which success criteria were met based on the findings of the evaluation, providing insights into the framework's strengths and areas for improvement.

Similar to the demonstration, we utilized the abstracted open source collaboration process for the evaluation. However, instead of using the JValue Hub as the artifact, we worked with the widely known tool GitHub. We selected GitHub as the tool for applying the framework on due to the participants' familiarity with its interface and collaboration processes, ensuring a consistent starting point and avoiding the need for tool-specific training. Furthermore, as a well-established platform for collaboration GitHub was well-suited for drawing comparisons to other potential application scenarios.

While the same overarching process was applied (figure 5.2), different steps of the process were used to suit the specific context of this evaluation.

6.1 Evaluation Objectives and Success Criteria

We evaluated the framework using the following success criteria, which we defined as detailed extensions of the overarching thesis objectives outlined in chapter 3. The goal of this evaluation is to determine whether these criteria are successfully met from the perspective of our assumed primary user group.

6.1.1 Evaluation Objectives

The primary objective of this evaluation is to assess the usability, effectiveness, accessibility, and adaptability of the conceptual behavioral design framework developed in this thesis. Specifically, the evaluation seeks to determine how well the framework facilitates intuitive design for developers with little to no prior

experience in behavioral design or design. It aims to evaluate the framework's ability to make behavioral design concepts accessible and actionable for these users, particularly in open source collaboration contexts.

Referring back to the objectives we defined in chapter 3, the evaluation focuses on the following key aspects:

1. **Usability:** How user-friendly, intuitive, and easy-to-learn the framework is for developers unfamiliar with behavioral design concepts. User satisfaction after applying the framework is also a critical indicator.
2. **Effectiveness:** The extent to which the framework achieves its intended outcomes — namely, enabling (behavioral) design improvements aligned with specific user or developer goals.
3. **Adaptability:** The framework's flexibility in addressing a range of use cases of different domains, particularly how well it supports customization to specific contexts or scenarios.
4. **Accessibility:** The framework's success in making behavioral design concepts and decision-making processes understandable and approachable for non-designers, especially those reluctant to engage with unfamiliar design principles.

6.1.2 Success Criteria

To evaluate the framework against these objectives, a set of success criteria and corresponding indicators was established. We aimed to comprehensively evaluate various aspects of the framework's performance, through assessing the criteria of its usability, effectiveness, adaptability, and accessibility. A table summarizing our defined success criteria and their corresponding success indicators can be seen in table 6.1.

Criterion	Success Indicator
Usability	
User-Friendliness	The framework should be easy to navigate and apply for developers with minimal design experience.
Intuitiveness	The framework should be logical and self-explanatory, with minimal confusion or frustration.
Learnability	Participants should understand core concepts after a brief introduction or one session of use.
Satisfaction	Users should report positive feedback, indicating value added to their workflow.
Effectiveness	
Design Improvements	The framework should guide users to actionable, contextually relevant design solutions aligned with behavioral design principles.
Goal Alignment	Outcomes achieved should align with the predefined goals of the design or process being improved.
Accessibility	
Understandability	The framework should clearly communicate behavioral design concepts in ways that are easy to grasp for non-designers.
Approachability	The framework should reduce the intimidation or reluctance felt by developers unfamiliar with behavioral design, encouraging their active engagement.
Clarity of Process	Decision-making processes and workflows within the framework should be clearly outlined and easy to follow without excessive guidance or trial and error.
Adaptability	
Customizability	The framework should allow for easy adjustments to suit specific use cases or project goals.
Ease of Modification	Modifications or updates should require minimal time and impact a small number of framework components.

Table 6.1: Overview of the success criteria and associated indicators

The framework's usability was a primary focus, as it needed to be accessible and practical for developers with varying levels of design experience.

Effectiveness focused on how well the framework facilitated actionable solutions that were aligned with the provided behavioral design principles. Adaptability was assessed in terms of the framework's flexibility to accommodate different use cases and project goals. Accessibility focused on ensuring the framework is inclusive, enabling a wide range of users, regardless of their prior expertise or background, to engage with and apply its principles effectively.

6.2 Evaluation Method and Process

The following chapter outlines the methodology used to evaluate the framework. It then delves into the evaluation process itself, detailing how the framework was assessed and describing the specific scenarios in which it was applied.

6.2.1 Evaluation Method

To evaluate the framework, we conducted a Think-Aloud Protocol with five participants. Each participant was given the same series of specific process steps of a use case to apply the NUDGE framework on while verbalizing their thoughts, actions, and decision-making processes in real-time. Recognizing shifts in their voice and analyzing the interviews, we gained in-depth insights into the participants' cognitive processes, helping to identify usability challenges, areas of confusion, and the framework's overall intuitiveness. Observations and verbal feedback were documented and later analyzed to identify patterns, common issues, and potential improvements.

The Think-Aloud Protocol was particularly suitable for this research as it aligned with assessing and evaluating the framework's practical application by non-design experts, one of the objectives defined in chapter 3. By observing participants' real-time interactions, it is possible to assess its accessibility, clarity, and effectiveness in guiding design decisions based on the participants' reactions (Eccles and Arsal, 2017; Fonteyn et al., 1993).

6.2.2 Evaluation Process

For the evaluation we presented participants with different scenarios tied to GitHub's collaboration process. Each of these scenarios focused on achieving a specific goal using the NUDGE framework. For the evaluation we used the framework, which we had previously adapted to the open source collaboration context for the demonstration.

Referring back to the polar sampling we did for the selection of process steps within the demonstration of the framework in chapter 5.1, we used the same matrix to choose fitting steps for the evaluation. So that in combination of the

steps used in the demonstration and the ones used for the evaluation, we could cover the framework comprehensively. Two process steps of varying complexity were selected via this polar sampling approach to ensure the inclusion of diverse scenarios and techniques. Similar to the demonstration's selection, these steps captured both simple and complex tasks to test the framework's flexibility and depth.

The evaluation involved five participants, all experienced computer scientists and developers with extensive expertise in application development but limited exposure to design and behavioral design. This selection aligned with the framework's target audience of developers who may lack formal training in design but aim to apply behavioral design principles effectively. The participants' familiarity with GitHub, an established tool for code collaboration, ensured a consistent starting point for all sessions, avoiding the need for tool-specific training and preventing misunderstandings stemming from unfamiliarity.

As explained in the former solution design chapter 4, the complete process of using the framework is done in two steps. The first step involves adapting the framework to a specific use case, as was done for the open source collaboration process in the demonstration chapter 5. The second step involves applying the framework's suggestions to the product or artifact being optimized, according to the previously formulated developer intents.

Because the first step, the adaptation phase, is very time-intensive, the evaluation within this master's thesis focused only on the second step, the application phase. This focus allowed for better comparability of results across different evaluation sessions, since all participants worked with the same developer intents and contents of the adapted framework. It required less time, and could therefore yield tangible changes in the artifact within just one session per participant - therefore participants were able to see the final outcomes of their implementations in the artifact and could assess for their value.

We utilized the adapted framework developed during the demonstration chapter 5 for the open source collaboration process, as GitHub facilitates this process for project collaboration and had already been integrated into the abstraction process used in the adapted framework.

Overall, the evaluation process consisted of five individual sessions. Between sessions, the framework was adjusted to address significant usability issues immediately, ensuring these did not obstruct the following sessions or the identification and resolution of other challenges within the framework. This iterative process not only enhanced the artifact incrementally but also ensured a smoother evaluation process for subsequent sessions. The framework's iterative refinement allowed for progressive improvement of the artifact throughout the evaluation. An excerpt showing how the framework appeared at the beginning of the evalu-

ation sessions and how it evolved in later versions can be found in chapter 6.5. A more detailed explanation of the changes we made to the framework, the underlying rationale, and why it developed in specific directions is also provided there.

Session Design and Structure To apply the framework, participants were presented with two steps of the abstracted open source collaboration process (figure 5.2) and therefore also GitHub's collaboration process on which they were to apply the framework to achieve a specific, predefined goal. These steps in combination with their intended goals and their representation through GitHub's user interface will henceforth be referred to as scenarios.

Participants were to use the NUDGE framework adapted to the open source collaboration process, shown in chapter A.1, figure 1. A Miro board was shared as the workspace. On this board, the task description was written out. Additionally, the abstracted open source collaboration process itself was displayed, with the selected steps to be addressed marked to provide better context and positioning. Screenshots of the GitHub user interface of the presented process steps were also available on the board, on which participants were to work and implement their design changes on. An overview of the setup can be found in the appendix chapter A.2 in figure 2.

To observe the participants' interaction with the framework, participants accessed both the board and the framework, still in the form of a table shared via Google Sheets at this stage, and shared their screen. This allowed us to observe the process alongside the participants' Think-Aloud commentary. Observations focused not only on the content of what was said but also on where participants lingered, how often they switched their focus between the framework and the task description on the Miro board, and similar behaviors.

Each session began with an introductory briefing to provide participants with context and align their understanding of the framework and its intended use. This included a demonstration of an example scenario, complete with a sample implementation, to familiarize participants with the process of implementing changes in the context of the session. Afterwards, we addressed questions to clarify any uncertainties before the tasks commenced.

Participants were then given time to read the task description themselves. After they indicated readiness, any further questions were addressed. Once everything was clear, participants were informed that urgent questions could still be raised during the task but that we as the observers would otherwise refrain from intervening, leaving the participants to work independently.

Each of the two scenarios was allocated 15 minutes, though this served more as a guideline than a strict limit. Participants who finished early could simply

move on to the next scenario. If a participant exceeded the 15-minute mark, we provided a gentle reminder of the time and encouraged them to wrap up their current implementation within the next few minutes.

The Scenarios The scenarios used in the sessions were carefully structured to progressively introduce complexity, leveraging learning effects. At the beginning of each session, an example scenario was presented alongside a sample implementation to provide a clear starting point. Participants were then tasked with working on two scenarios: one simpler and one more complex. The simpler scenario was conducted first, closely resembling the example scenario, to allow participants an easier entry point and to reserve sufficient cognitive resources for becoming acquainted with the framework.

Example Scenario: Initiating Choice for Forking The example scenario focused on the step *Initiate Choice for Forking*. The objective was to encourage users to fork a project as an initial step in the collaboration process.

Scenario 1: Initiate Choice for Contribution The first scenario handled by the participants was of the step *Initiate Choice for Contribution*. It was very similar to the example scenario with only a different objective to nudge users towards making the choice to contribute to a project. This scenario had a very small scope regarding the achievement of the predefined goal and was straightforward to execute. It served to familiarize participants with the foundational elements of the framework.

Scenario 2: Create Contribution Request The second scenario was more complex and involved a multi-step process: *Create Contribution Request*. It presented a wide range of solution possibilities due to its broader scope and allowed for a deeper exploration of the framework's application.

6.3 Evaluation Results

The following results are based on the Think-Aloud sessions we conducted with participants. We begin by presenting some numerical insight to provide a quantitative overview of how the sessions unfolded. This is followed by a comprehensive qualitative analysis, which examines key themes, insights, and challenges associated with the application of the framework, as well as scenario-specific feedback.

6.3.1 Numerical Insights

We derived the following figures from our notes of the evaluation sessions. These numbers provide insights into the progression of the sessions, commonalities

between them, and notable differences.

Due to the qualitative nature of the evaluation, quantitative data was not the primary focus. However, these numerical observations highlight some usage patterns and recurring feedback trends:

Metric	**Observation**
Average Completion Time per Scenario	Scenario 1: ~15 minutes; Scenario 2: ~18 minutes
Frequency of Mentioned Techniques	Tunneling: 5/5 sessions; Tailoring: 4/5 sessions; Simulation: 3/5 sessions
Instances of Overlapping Techniques	Mentioned in 4/5 sessions; personalization and tailoring most frequently cited as overlapping
Average Number of Suggestions per Participant	4-5 distinct suggestions per session

Table 6.2: Numerical insights of the evaluation sessions

6.3.2 Qualitative Data

Thematic analysis of participants' feedback revealed several key themes, insights, and challenges with the framework's usability and application. First, we present general insights, followed by insights specific to each scenario.

Key Themes and Insights Participants viewed the framework as a flexible "mental framework" rather than a rigid tool. They appreciated its guidance for structuring their thought processes without enforcing prescriptive rules. Techniques such as tunneling, tailoring, simulation, and reduction were particularly valued for their clarity and applicability, standing out as practical and accessible elements within the framework.

Through applying the framework, participants observed clear improvements in their perceived user experience. These included a sharper focus on primary actions and a reduction in complexity by tailoring information to user expertise. The framework facilitated thought processes about design and user experience aspects that might not have been explicitly considered otherwise, functioning well as a kind of checklist for design considerations and providing direction for structured idea generation.

The framework demonstrated effectiveness by enabling participants to generate actionable, context-specific design ideas. Improvements included increasing the visibility of key contribution actions and tailoring interfaces for novice users. The framework also served as a valuable prompt for structured thinking towards

design strategies, particularly benefiting participants unfamiliar with behavioral design principles.

During the evaluation sessions, the techniques tunneling and tailoring were the most frequently mentioned and applied by participants. Their repeated use across scenarios indicated their clarity and practicality, making them particularly useful for guiding participants in improving user experience.

Challenges in Application Participants often found the framework text-heavy, leading to cognitive overload. Some struggled to distinguish between overlapping techniques like tailoring and personalization. Several participants also noted that some framework elements seemed redundant, particularly some developer intents that were displayed multiple times and were mapped to different techniques. They were reported to be worded similarly or identically, compounding the confusion. Participants also expressed uncertainty about whether using a mixture of techniques or fluid transitions between them were acceptable. The accessibility of the framework was another area of mixed feedback. Participants appreciated the examples provided but emphasized a need for more concise and clear guidance to make the framework easier to use, particularly for those new to behavioral design concepts and within the first use of the framework.

Scenario-Specific Feedback In comparing the scenarios, participants found scenario 1 to be easier to address due to its narrower scope and simpler tasks. In contrast, scenario 2 presented greater complexity, but leveraged learning effects of the implementations within the first scenario. However, it also highlighted redundancies and repetitive information in the framework itself. This occasionally distracted participants, requiring more effort to navigate and apply the framework effectively.

Scenario 1: Simpler and Exploratory Scenario 1 was encouraging participants to explore and ideate without overloading them at first use. Given the limited scope and interaction possibilities of scenario 1, participants gravitated toward techniques such as tailoring and personalization to adapt the user interface's context based on user expertise. They identified opportunities to show more explanatory content for novice users and reduce distractions for experienced users.

Participants regularly approached scenario 1 with a focus on highlighting key actions for contribution. They considered how to emphasize overlooked elements, reduce irrelevant visual elements, and motivate user contributions through contextual cues.

Specific Interventions:

- Participants emphasized overlooked elements such as the ”contribute” button, using the tunneling technique's visual hierarchy adjustments to make them more prominent.
- They reduced the prominence of elements irrelevant to the user's immediate intent, such as deemphasizing the ”code” button when it was not the primary focus.
- All participants made changes to reposition key buttons, moving them to more central locations for greater visibility.
- They suggested motivational tools, such as showcasing contributors or highlighting successful pull requests, to foster a sense of community and encourage engagement.

Scenario 2: Complex but Focused Scenario 2 was more challenging due to the fact that the process step this covered, *create contribution request*, was of higher complexity and a multi-task process step. While this required participants to invest more effort, it leveraged learning effects and allowed for the application of repeatable patterns identified in scenario 1. The larger and more complex scenario allowed for a wider range of varied implementations among the participants. The increased complexity also led to more nuanced changes, particularly in simplifying processes and addressing the varied needs of novice and experienced users.

Participants typically focused on simplifying the user journey and breaking down complex workflows into manageable steps. They applied techniques such as reduction to streamline the process and tailoring to adjust the experience for users of different expertise levels. The learning effects from scenario 1 were evident, as participants applied similar strategies with greater precision in scenario 2.

Specific Interventions:

- Participants incorporated additional steps into the existing process with visual progress indicators to simplify complex workflows, aligning with the reduction and self-monitoring technique.
- Tailoring was introduced through contextual explanations aimed at novice users, combined with options for experienced users to skip onboarding content.
- Additional recommendations for novice users included explanatory tooltips.

- Participants suggested adding summary views aligned with the rehearsal and simulation techniques, allowing users to review their steps before finalizing actions.

Overall, scenario 1 provided an opportunity for participants to focus on exploratory changes with immediate visual impact, while scenario 2 required them to tackle more systemic improvements.

6.3.3 Alignment with Success Criteria

The evaluation of the framework demonstrated mostly positive outcomes in relation to the success criteria, highlighting strengths as well as areas for improvement.

Objectives and Success Criteria	**Met**
Usability	
User-friendliness	Yes
Intuitiveness	Partially
Learnability	Yes
Satisfaction	Yes
Effectiveness	
Design improvements	Yes
Goal alignment	Yes
Accessibility	
Understandability	Partially
Approachability	Yes
Clarity of process	Partially
Adaptability	Partially

Table 6.3: Assessment of met objectives and success criteria

Usability In terms of usability, the framework proved effective in facilitating ideation and enabling participants to generate meaningful design improvements.

- **User-friendliness:** Even with little to no prior exposure to behavioral design, participants were able to apply the framework effectively, meeting this criterion.
- **Intuitiveness:** This criterion was only partially met. Some points of confusion arose, particularly in the earlier versions of the framework during the first scenario. Issues such as the redundancy of developer intents required additional clarification.
- **Learnability:** The framework successfully met this criterion, as all participants demonstrated a clear understanding of its core concepts. A learning

effect was also observed, with participants becoming more proficient over time.

- **Satisfaction:** This criterion was met most successfully. After their sessions, the framework exceeded participants' earlier expectations in terms of usefulness, prompting creative and contextually relevant suggestions. Many participants were pleasantly surprised by their ability to generate innovative ideas, showcasing the framework's potential to unlock new perspectives, even for those without a design background.

Effectiveness The framework successfully met our success criterion of effectiveness by guiding participants in identifying design improvements through behavioral design principles that aligned with their goals.

- **Design improvements** were observed in all sessions and by all participants. The framework effectively facilitated the generation of contextually relevant design solutions, which were implemented successfully.
- **Goal alignment** was also achieved, as participants followed the developer intents and implemented changes that fulfilled the scenario-specific goals.

Accessibility In terms of accessibility, the framework effectively introduced behavioral design concepts to non-experts, lowering barriers to entry. However, participants noted that further refinements could enhance its accessibility, making the concepts even more actionable for users with limited prior design experience.

- **Understandability:** This criterion was only partially met. While the framework provided clear communication of its behavioral design concepts, some participants found the explanations to be too brief and expressed a desire for more detailed guidance.
- **Approachability:** This criterion was successfully met. The framework's mapping of appropriate techniques to specific process steps encouraged participants to actively engage with the different behavioral design strategies.
- **Clarity of process:** This criterion was mostly met. Participants found the process itself to be clear; however, since they did not adapt the framework to the use case themselves and instead worked with predefined developer intents, some confusion arose regarding the intent behind certain elements.

Adaptability Of the two phases of using the framework — the adaptation phase and the application phase — this evaluation focused solely on the application phase. For the evaluation, we used the version of the framework that had already been adapted as part of the demonstration. As a result, we cannot make definitive statements about how adaptable the framework is to other use cases.

Although the framework's structure and design provides adaptability for use cases that can be formulated as formal processes, further research is needed to explore and evaluate how effectively it adapts to different contexts, particularly in terms of its usefulness.

6.4 Limitations of the Evaluation

While the evaluation process provided valuable insights into the usability, effectiveness, accessibility and adaptability of the framework, some constraints and limitations may have impacted the findings.

Sample Size The evaluation involved five participants, a relatively small sample size that limits the statistical generalizability of the results. While the qualitative insights gathered were detailed, a larger and more diverse group of participants would provide stronger evidence for the framework's usability and accessibility across different user groups.

Participant Background and Bias Participants were all experienced developers with high expertise in application development but limited exposure to behavioral design. While this aligns with the primary target audience of the framework, it introduces potential biases, such as:

- **Familiarity Bias:** Participants' prior knowledge of tools like GitHub might have influenced their ability to generate ideas independently of the framework, skewing the evaluation of its effectiveness.
- **Acquaintance bias:** Since participants knew the evaluator personally, their responses might have been influenced by a desire to provide positive feedback.

These factors may limit the applicability of findings to developers with differing levels of experience or those unfamiliar with GitHub's workflows and interface.

Time Constraints The evaluation sessions were time-limited to approximately 15-18 minutes per scenario, which may not have been sufficient for participants to fully explore the framework's capabilities. As a result, some techniques or deeper insights might have been overlooked due to time pressure.

Framework Complexity Participants reported that the framework felt text-heavy and redundant. This complexity might have hindered their ability to use it effectively within the constrained evaluation period. As a result, usability issues identified in the evaluation may partially reflect the testing conditions and the limited time frame rather than inherent flaws in the framework. In practical real

life applications of the framework, it is unlikely that time constraints will be as stringent.

Iterative Changes to the Framework The framework was iteratively updated between sessions based on feedback from earlier participants. While this iterative process improved the artifact, it introduced variability across sessions, as participants interacted with slightly different versions of the framework. This limits the consistency of the findings and makes it challenging to isolate the impact of specific changes.

6.5 The Framework's Evolution through its Evaluation

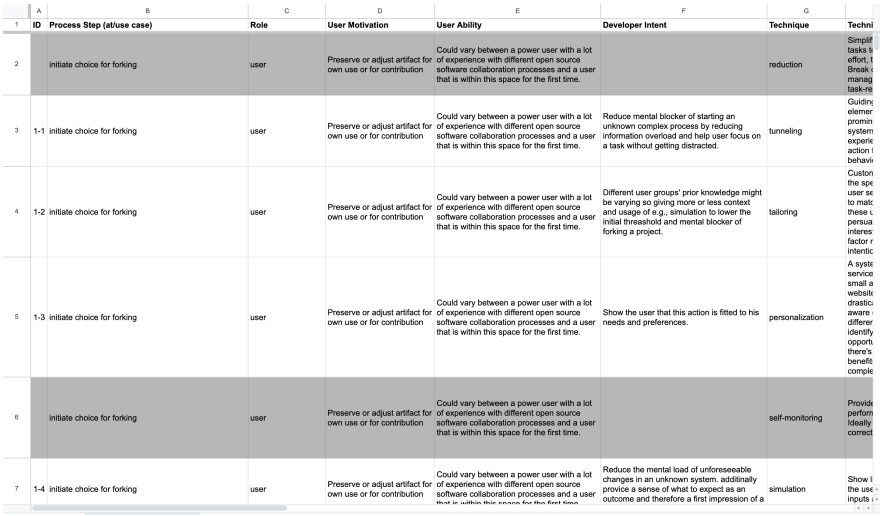
This section provides a brief overview of how the appearance and structure of the NUDGE framework evolved based on feedback from the evaluation process.

The framework was initially designed as a table, with developer intents in combination with the technique and the process step serving as the key for each row and can be seen in figure 6.1. At this stage it was supposed to be used more as a manual or handbook, where developers could find inspiration if needed. The rows' associated columns included details such as user abilities, the relevant process step and mapped behavioral technique, and specific details of that technique, such as its summary. The first version of the framework also featured a column for prioritizing specific rows, allowing developers adapting the framework to highlight the most promising techniques for a given process step.

However, this prioritization column proved problematic. Participants often overlooked it or found it confusing, as their own priorities frequently differed from those presented in the framework. This misalignment left participants unsure about their technique choices. As a result, the prioritization column was removed in version 2.

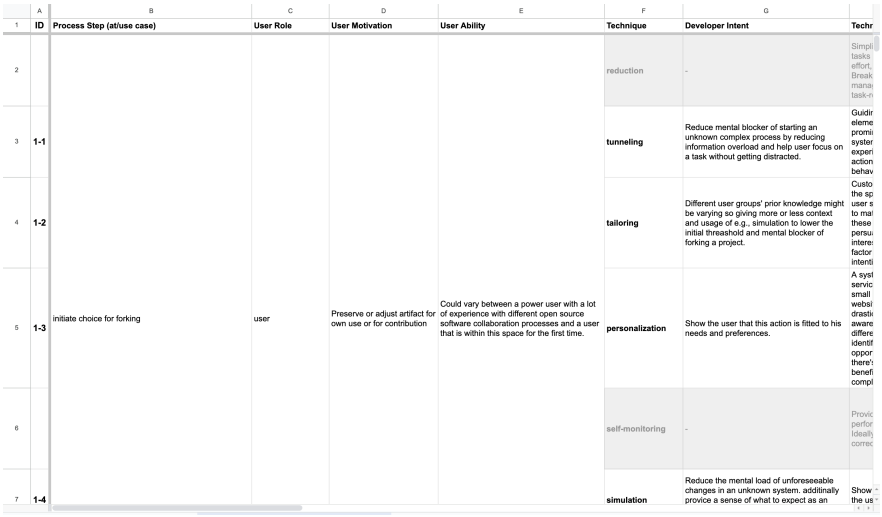
Even without the prioritization column, some participants still found the table's structure confusing due to redundant information. For example, the process step cell was repeated across multiple rows since it was associated with multiple developer intents and techniques, creating unnecessary clutter. In version 3 (figure 6.2), we attempted to address this by consolidating redundant cells, such as combining the "process step" cell of rows where a single process step was mapped to multiple behavioral techniques

Despite these changes, some usability issues persisted. Some participants struggled to view the framework as a cohesive whole and were overwhelmed by the remaining redundancy. To resolve these challenges, we completely reimagined the frame-



	A	B	C	D	E	F	G	
1	ID	Process Step (at/use case)	Role	User Motivation	User Ability	Developer Intent	Technique	Techni
2		initiate choice for forking	user	Preserve or adjust artifact for own use or for contribution	Could vary between a power user with a lot of experience with different open source software collaboration processes and a user that is within this space for the first time.		reduction	Simplif tasks b effort, t Break manag task-re
3	1-1	initiate choice for forking	user	Preserve or adjust artifact for own use or for contribution	Could vary between a power user with a lot of experience with different open source software collaboration processes and a user that is within this space for the first time.	Reduce mental blocker of starting an unknown complex process by reducing information overload and help user focus on a task without getting distracted.	tunneling	Guidin elemer promin system experi action behavi
4	1-2	initiate choice for forking	user	Preserve or adjust artifact for own use or for contribution	Could vary between a power user with a lot of experience with different open source software collaboration processes and a user that is within this space for the first time.	Different user groups' prior knowledge might be varying so giving more or less context and usage of e.g., simulation to lower the initial threashold and mental blocker of forking a project.	tailoring	Custon the spe user se to mat these persua interes factor intentic
5	1-3	initiate choice for forking	user	Preserve or adjust artifact for own use or for contribution	Could vary between a power user with a lot of experience with different open source software collaboration processes and a user that is within this space for the first time.	Show the user that this action is fitted to his needs and preferences.	personalization	A syste service small a websit drastic aware differe identify opport there's benefit comple
6		initiate choice for forking	user	Preserve or adjust artifact for own use or for contribution	Could vary between a power user with a lot of experience with different open source software collaboration processes and a user that is within this space for the first time.		self-monitoring	Provid perfor Ideally correct
7	1-4	initiate choice for forking	user	Preserve or adjust artifact for own use or for contribution	Could vary between a power user with a lot of experience with different open source software collaboration processes and a user that is within this space for the first time.	Reduce the mental load of unforeseeable changes in an unknown system. additinally provice a sense of what to expect as an outcome and therefore a first impression of a	simulation	Show the use inputs

Figure 6.1: NUDGE framework version 1-2



	A	B	C	D	E	F	G	
1	ID	Process Step (at/use case)	User Role	User Motivation	User Ability	Technique	Developer Intent	Techr
2						reduction	-	Simpli tasks effort, Break manag task-r
3	1-1					tunneling	Reduce mental blocker of starting an unknown complex process by reducing information overload and help user focus on a task without getting distracted.	Guidir eleme promi syste experi action behav
4	1-2					tailoring	Different user groups' prior knowledge might be varying so giving more or less context and usage of e.g., simulation to lower the initial threashold and mental blocker of forking a project.	Custo the sp user s to ma these persua interes factor intenti
5	1-3	initiate choice for forking	user	Preserve or adjust artifact for own use or for contribution	Could vary between a power user with a lot of experience with different open source software collaboration processes and a user that is within this space for the first time.	personalization	Show the user that this action is fitted to his needs and preferences.	A syst servic small websi drasti aware differe identif oppor there's benefi compl
6						self-monitoring	-	Provic perfor Ideally correc
7	1-4					simulation	Reduce the mental load of unforeseeable changes in an unknown system. additinally provice a sense of what to expect as an	Show the us

Figure 6.2: NUDGE framework version 3

work's structure in version 4, introducing a three-dimensional visualization that maps process steps to behavioral design techniques through developer intents on a visual level as well. This new format aims to provide a clearer, more intuitive overview of the framework's structure and functionality and can be seen in detail in chapter A.1, figure 1.

6.6 Summary

The evaluation of the behavioral design framework demonstrated its potential to enhance usability and effectiveness in open source collaboration contexts, while also identifying areas for refinement. This section summarizes the key takeaways and recommendations for improvement derived from the evaluation process.

Key Takeaways

1. **Fulfilment of Objectives:** The framework successfully addressed most of its primary objectives, including usability, effectiveness, and accessibility. It provided developers an actionable tool to enhance behavioral design practices in their workflows. Participants were able to generate contextually relevant design improvements and align outcomes with predefined goals, validating the framework's intended purpose.
2. **Success Criteria:**
    - **Usability:** The framework was user-friendly and learnable, with participants reporting mostly positive experiences and satisfaction. However, redundancies and text-heavy content occasionally caused cognitive overload, which affected the intuitiveness criterion.
    - **Effectiveness:** Participants consistently identified actionable design improvements, aligning with behavioral design principles and fulfilling this evaluation goal.
    - **Accessibility:** The framework lowered barriers for non-designers, though additional refinement is needed to simplify explanations and reduce perceived complexity.
    - **Adaptability:** While we assume the framework to be adaptable to different domains, the framework's application phase wasn't considered in this evaluation and could therefore not be evaluated.
3. **Methodological Insights:** Using a Think-Aloud Protocol revealed the framework's strengths, such as its structured approach to design ideation, while also surfacing challenges like participants' initial confusion with redundant content.

4. **Framework Evolution:** The framework evolved significantly during the evaluation, transitioning from a text-heavy table format to a more visual, intuitive structure. These changes addressed usability issues and aim to increase clarity.

Recommendations for Improvement

1. **Simplify Content:** Reduce text-heavy explanations and streamline redundant information to improve accessibility and reduce cognitive load.
2. **Enhance Guidance:** Provide clearer instructions and more concise examples to support first-time users in understanding and applying the framework.
3. **Broader Applicability:** Conduct evaluations in diverse domains beyond open source collaboration to assess generalizability.

The evaluation highlighted the framework's strengths in guiding developers toward effective behavioral design solutions, while also identifying areas for improvement to enhance usability and scalability. In the context of open source collaboration, the framework appeared to be particularly valuable, as the majority of participants responded positively, noting that the changes it facilitated were both impactful and unexpected. The evaluation's findings provide a foundation for future refinements and broader adoption of the framework, advancing the integration of behavioral design into development practices.

7 Conclusion

The NUDGE framework marks a meaningful progression in connecting behavioral design principles with developer-focused workflows. By addressing challenges in creating intuitive, user-focused solutions in tech-heavy environments, it empowers developers to enhance usability without requiring formal design training. As a structured tool, the NUDGE framework enables developers to apply behavioral design strategies effectively and goal-oriented, helping them overcome obstacles in user behavior and design. Drawing from Fogg's Behavior Model and persuasive technology strategies (Fogg, 2003, 2009), this framework is designed to be both practical and accessible.

Core elements of the NUDGE Framework include

- its conceptual design, which ensures that the adaptation and application phases can be tailored to specific use cases while remaining adaptable to evolving contexts.
- its emphasis on process customization and developer intent mapping. By defining a clear, measurable process and formulating goals and intentions for each step through developer intents, the framework's effectiveness is significantly enhanced.
- its concept and behavioral blocks, containing techniques based on Fogg's persuasive technology strategies that offer practical action plans for addressing user behavior (Fogg, 2003).

While the framework's contributions are promising, there are limitations to consider, such as scalability issues in highly complex systems and an initial time invest for the frameworks adaptation to a use case. Future work would benefit from expanding the behavioral techniques library to incorporate emerging insights from the field of behavioral design, allowing the framework to evolve alongside the ecosystem. Enhancing scalability for larger, more complex systems would also be valuable. The NUDGE framework currently operates only on a conceptual level. Hands-on tools could simplify it's adaptation and application and improve its usability. Additionally, testing the framework across various contexts

and industries would help assess its generalizability beyond the tested contexts and refine its design.

In its application to the JValue platform, the NUDGE Framework has shown promise in addressing usability challenges and identifying the right behavioral strategies to resolve them. The positive evaluation results further emphasize its usefulness in developer-driven environments, particularly in the open source collaboration space. These findings suggest that the NUDGE Framework holds potential for making behavioral design principles more accessible and actionable for developers.

Integrating behavioral design into development processes plays a key role in creating intuitive, user-centered solutions. The NUDGE Framework illustrates how developers can apply structured methodologies to enhance usability and influence user behavior. By bridging the gap between design and development, it supports the creation of collaborative, accessible, and effective solutions, contributing to the advancement of behavior-driven design in development workflows.

Appendices

A Demonstration and Evaluation Resources

A.1 The NUDGE Framework Adapted to the Open Source Collaboration Use Case

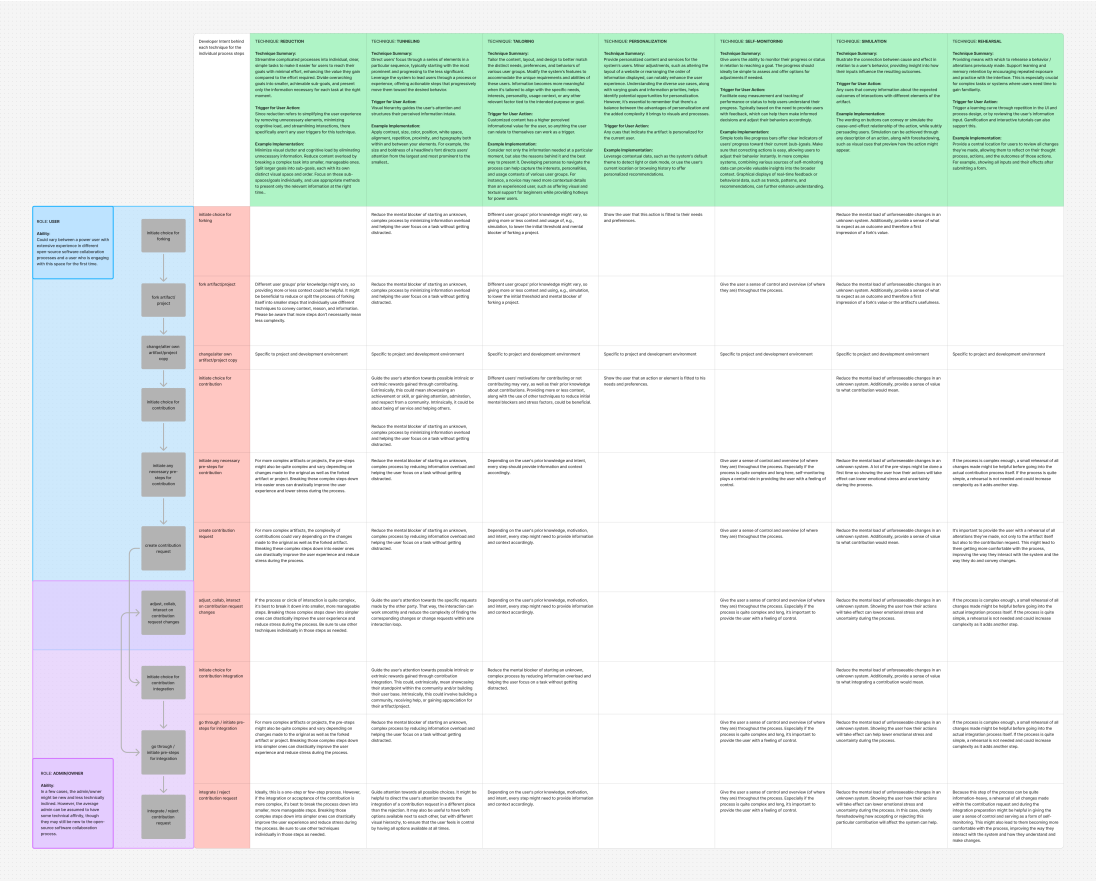


Figure 1: The open source collaboration use case adapted NUDGE framework, used for the demonstration and evaluation of the framework

A.2 The Evaluation Session Setup

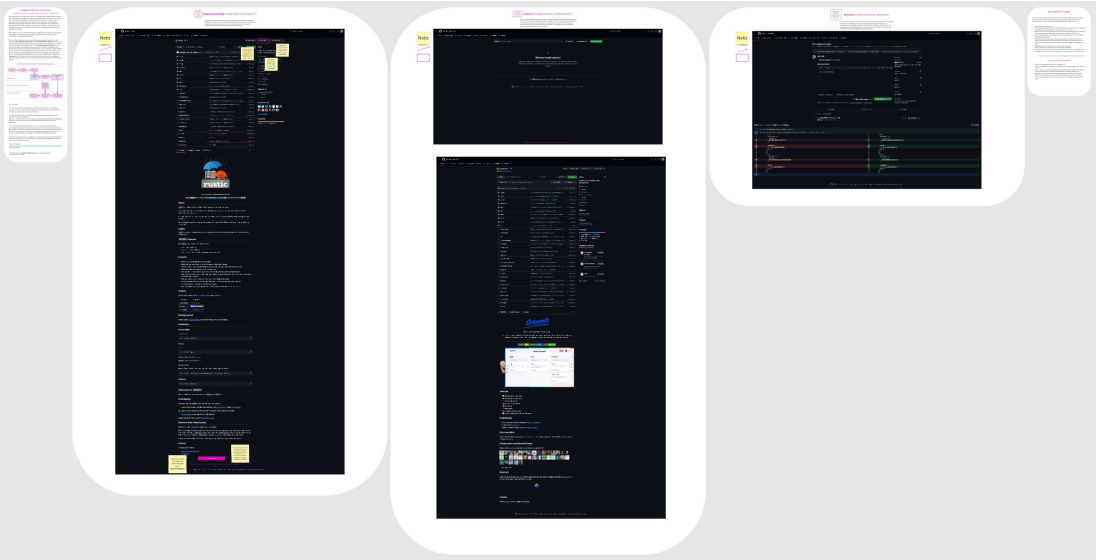


Figure 2: A high-level overview of the evaluation session setup

B Overview of the Base Set of Behavioral Blocks

B.1 Technique: Reduction

Technique Summary: Streamline complicated processes into individual, clear, simple tasks to make it easier for users to reach their goals with minimal effort, enhancing the value they gain compared to the effort required. Divide overarching goals into smaller, achievable sub-goals, and present only the information necessary for each task at the right moment.

Trigger for User Action: Since reduction refers to simplifying the user experience by removing unnecessary elements, minimizing cognitive load, and streamlining interactions, there specifically aren't any user triggers for this technique.

Example Implementation: Minimize visual clutter and cognitive load by eliminating unnecessary information. Reduce content overload by breaking a complex task into smaller, manageable ones. Split larger goals into sub-goals, each with its own distinct visual space and order. Focus on these sub-spaces/goals individually, and use appropriate methods to present only the relevant information at the right time.

B.2 Technique: Tunneling

Technique Summary: Direct users' focus through a series of elements in a particular sequence, typically starting with the most prominent and progressing to the less significant. Leverage the system to lead users through a process or experience, offering actionable steps that progressively move them toward the desired behavior.

Trigger for User Action: Visual hierarchy guides the user's attention and structures their perceived information intake.

Example Implementation: Apply contrast, size, color, position, white space, alignment, repetition, proximity, and typography both within and between your elements. For example, the size and boldness of a headline's font directs users' attention from the largest and most prominent to the smallest.

B.3 Technique: Tailoring

Technique Summary: Tailor the content, layout, and design to better match the distinct needs, preferences, and behaviors of various user groups. Modify the system's features to accommodate the unique requirements and abilities of these

users. Information becomes more meaningful when it's tailored to align with the specific needs, interests, personality, usage context, or any other relevant factor tied to the intended purpose or goal.

Trigger for User Action: Customized content has a higher perceived informational value for the user, so anything the user can relate to themselves can work as a trigger.

Example Implementation: Consider not only the information needed at a particular moment, but also the reasons behind it and the best way to present it. Developing personas to navigate the process can help capture the interests, personalities, and usage contexts of various user groups. For instance, a novice may need more contextual details than an experienced user, such as offering visual and textual support for beginners while providing hotkeys for power users.

B.4 Technique: Personalization

Technique Summary: Provide personalized content and services for the system's users. Minor adjustments, such as altering the layout of a website or rearranging the order of information displayed, can notably enhance the user experience. Understanding the diverse use cases, along with varying goals and information priorities, helps identify potential opportunities for personalization. However, it's essential to remember that there's a balance between the advantages of personalization and the added complexity it brings to visuals and processes.

Trigger for User Action: Any cues that indicate the artifact is personalized for the current user.

Example Implementation: Leverage contextual data, such as the system's default theme to detect light or dark mode, or use the user's current location or browsing history to offer personalized recommendations.

B.5 Technique: Self-Monitoring

Technique Summary: Give users the ability to monitor their progress or status in relation to reaching a goal. The progress should ideally be simple to assess and offer options for adjustments if needed.

Trigger for User Action: Facilitate easy measurement and tracking of performance or status to help users understand their progress. Typically based on the need to provide users with feedback, which can help them make informed decisions and adjust their behaviors accordingly.

Example Implementation: Simple tools like progress bars offer clear indicators of users' progress toward their current (sub-)goals. Make sure that correcting actions is easy, allowing users to adjust their behavior instantly. In more complex systems, combining various sources of self-monitoring data can provide valuable insights into the broader context. Graphical displays of real-time feedback or behavioral data, such as trends, patterns, and recommendations, can further enhance understanding.

B.6 Technique: Simulation

Technique Summary: Illustrate the connection between cause and effect in relation to a user's behavior, providing insight into how their inputs influence the resulting outcomes.

Trigger for User Action: Any cues that convey information about the expected outcomes of interactions with different elements of the artifact.

Example Implementation: The wording on buttons can convey or simulate the cause-and-effect relationship of the action, while subtly persuading users. Simulation can be achieved through any description of an action, along with foreshadowing, such as visual cues that preview how the action might appear.

B.7 Technique: Rehearsal

Technique Summary: Providing means with which to rehearse a behavior / alterations previously made. Support learning and memory retention by encouraging repeated exposure and practice with the interface. This is especially crucial for complex tasks or systems where users need time to gain familiarity.

Trigger for User Action: Trigger a learning curve through repetition in the UI and process design, or by reviewing the user's information input. Gamification and interactive tutorials can also support this.

Example Implementation: Provide a central location for users to review all changes they've made, allowing them to reflect on their thought process, actions, and the outcomes of those actions. For example, showing all inputs and their effects after submitting a form.

References

Ashley, J., & Desmond, K. (2005). Success with user-centered design management. *Interactions*, *12*(3), 27–32. https://doi.org/10.1145/1060189.1060211

Bay Brix Nielsen, C. K. E., Cash, P., & Daalhuizen, J. (2024). The power and potential of behavioural design: Practice, methodology, and ethics. *Journal of Engineering Design*, *35*(5), 504–542. https://doi.org/10.1080/09544828.2024.2322897

Bucher, A. (2020). *Engaged: Designing for behavior change*. Rosenfeld Media.

Cash, P., Vallès, X., Echstrøm, I., & Daalhuizen, J. (2022). Method use in behavioural design: What, how, and why? *International Journal of Design*. https://doi.org/10.57698/V16I1.01

Cash, P. J., Hartlev, C. G., & Durazo, C. B. (2017). Behavioural design: A process for integrating behaviour change and design. *Design Studies*, *48*, 96–128. https://doi.org/10.1016/j.destud.2016.10.001

Chiang, I.-Y., Lin, P.-H., Kreifeldt, J. G., & Lin, R. (2021). From theory to practice: An adaptive development of design education. *Educ. Sci. (Basel)*, *11*(11), 673.

Chopra, A. K., Mylopoulos, J., Dalpiaz, F., Giorgini, P., & Singh, M. P. (2010). Requirements as goals and commitments too. In *Intentional perspectives on information systems engineering* (pp. 137–153). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-12544-7_8

Chung, L., Nixon, B. A., Yu, E., & Mylopoulos, J. (2000). *Non-functional requirements in software engineering*. Springer US. https://doi.org/10.1007/978-1-4615-5269-7

Eberhard, K. (2021). The effects of visualization on judgment and decision-making: A systematic literature review. *Management Review Quarterly*, *73*(1), 167–214. https://doi.org/10.1007/s11301-021-00235-8

Eccles, D. W., & Arsal, G. (2017). The think aloud method: What is it and how do i use it? *Qualitative Research in Sport, Exercise and Health*, *9*(4), 514–531. https://doi.org/10.1080/2159676x.2017.1331501

Fogg, B. (2003). *Persuasive technology*. Elsevier. https://doi.org/10.1016/b978-1-55860-643-2.x5000-8

Fogg, B. (2009). A behavior model for persuasive design. *Proceedings of the 4th International Conference on Persuasive Technology*. https://doi.org/10.1145/1541948.1541999

Fonteyn, M. E., Kuipers, B., & Grobe, S. J. (1993). A description of think aloud method and protocol analysis. *Qualitative Health Research*, *3*(4), 430–441. https://doi.org/10.1177/104973239300300403

Heltweg, P., & Riehle, D. (2023). A systematic analysis of problems in open collaborative data engineering. *ACM Transactions on Social Computing*, *6*(3–4), 1–30. https://doi.org/10.1145/3629040

Khadilkar, P. R., & Cash, P. (2020). Understanding behavioural design: Barriers and enablers. *Journal of Engineering Design*, *31*(10), 508–529. https://doi.org/10.1080/09544828.2020.1836611

Khaled, R., Noble, J., & Biddle, R. (2005). An analysis of persuasive technology tool strategies. *IWIPS*, 167–173.

Levesque, M. (2004). Fundamental issues with open source software development. *First Monday*, *9*(4). https://doi.org/10.5210/fm.v9i4.1137

Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning*, *1*(1), 47–80. https://doi.org/10.1023/a:1022691120807

Pal, S., Nair, A., & Zuo, Z. (2024). Collaborative dynamics in open source software development: Unveiling the influence of team interaction and the role of project manager. *Journal of Operations Management*, *70*(7), 1076–1099. https://doi.org/10.1002/joom.1324

Parsons, P., & Sedig, K. (2013). Common visualizations: Their cognitive utility. In *Handbook of human centric visualization* (pp. 671–691). Springer New York. https://doi.org/10.1007/978-1-4614-7485-2_27

Raza, A., & Capretz, L. F. (2015). Do open source software developers listen to their users? *First Monday: Peer-Reviewed Open Journal on the Internet*, *17*(3), 1–9. https://doi.org/10.48550/ARXIV.1507.06893

Stanford Behavior Design Lab. (2019). *Behavior design lab at stanford university* [Accessed on September 28, 2024. Year derived from last update on copyright.]. Retrieved September 28, 2024, from https://behaviordesign.stanford.edu/

Steinmacher, I., Conte, T., Gerosa, M. A., & Redmiles, D. (2015). Social barriers faced by newcomers placing their first contribution in open source software projects. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, 1379–1392. https://doi.org/10.1145/2675133.2675215

Toledo, F. P. d., Devincenzi, S., Kwecko, V., Mota, F. P., & Botelho, S. S. d. C. (2018). A framework for modeling persuasive technologies based on the fogg behavior model. *2018 IEEE Frontiers in Education Conference (FIE)*, 1–5. https://doi.org/10.1109/fie.2018.8659195

Tyler, R. W. (1966). The behavioral sciences and the schools. *Teachers College Record: The Voice of Scholarship in Education*, *67*(10), 200–214. https://doi.org/10.1177/016146816606701008

van Kuijk, J., Daalhuizen, J., & Christiaans, H. (2019). Drivers of usability in product design practice: Induction of a framework through a case study of three product development projects. *Design Studies*, *60*, 139–179. https://doi.org/10.1016/j.destud.2018.06.002

Voorheis, P., Zhao, A., Kuluski, K., Pham, Q., Scott, T., Sztur, P., Khanna, N., Ibrahim, M., & Petch, J. (2022). Integrating behavioral science and design thinking to develop mobile health interventions: Systematic scoping review. *JMIR mHealth and uHealth*, *10*(3), e35799. https://doi.org/10.2196/35799

Wildeboer, G., Kelders, S. M., & van Gemert-Pijnen, J. E. (2016). The relationship between persuasive technology principles, adherence and effect of web-based interventions for mental health: A meta-analysis. *International Journal of Medical Informatics*, *96*, 71–85. https://doi.org/10.1016/j.ijmedinf.2016.04.005

Wölbling, A., Krämer, K., Buss, C. N., Dribbisch, K., LoBue, P., & Taherivand, A. (2012). Design thinking: An innovative concept for developing user-centered software. In *Software for people* (pp. 121–136). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-31371-4_7