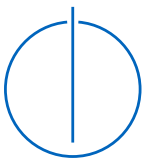
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

A Social and Interactive Tool to Support Pattern Communities in Large-Scale Agile Development

Murat Güner



DEPARTMENT OF INFORMATICS

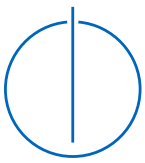
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

A Social and Interactive Tool to Support Pattern Communities in Large-Scale Agile Development

Ein soziales und interaktives Werkzeug zur Unterstützung von Mustergemeinschaften bei der skalierten agilen Softwareentwicklung

Author:	Murat Güner
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Ömer Uludağ
Submission Date:	15.04.2021



I confirm that this master's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.04.2021 Murat Güner

Acknowledgments

First and foremost, I would like to thank my supervisor Ömer Uludağ for his support during this research project. I learned many important things from you and always enjoyed our close collaboration.

Also, I would like to thank Professor Dr. Florian Matthes for the opportunity to write my thesis at his chair for Software Engineering for Business Information Systems (SEBIS).

Last but not least, I would like to thank my family and all my friends for their unconditional support during this exciting time.

Abstract

Today's business environment is very dynamic, and traditional software development methodologies cannot cope with unpredictable business environments and changing customer demands. To address these challenges, the agile movement emerged in the 1990s, leading to the development of Agile Manifesto and many agile software development methods, such as Extreme Programming and Scrum. Agile methods brought unprecedented changes to the software engineering field by accentuating change tolerance, customer involvement, and team collaboration. The success of agile methods for small, co-located teams has inspired companies to apply agile methods to large-scale projects. Since those methods are initially designed for small teams, novel challenges occur when introducing them at a larger scale, such as inter-team coordination and communication, dependencies with other organizational units, or general resistance to changes.

Since patterns provide structured solutions to recurring concerns, we aim to support pattern communities' establishment in large-scale agile development by addressing typical challenges of pattern languages.

List of Figures

List of Tables

Contents

1 Introduction

This chapter presents the motivation for this master's thesis by revealing the necessity of supporting the creation of a pattern community in the field of large-scale agile development by a social and interactive tool in Section 1.1. Following this, the objectives and the corresponding research questions of the thesis are highlighted in Section 1.2. The succeeding Section 1.3 describes the underlying research approach of the thesis.

1.1 Introduction

> Indeed, one of my major complaints about the computer field is that whereas Newton could say, "If I have seen a little farther than others, it is because I have stood on the shoulders of giants," I am forced to say, "Today we stand on each other's feet." Perhaps the central problem we face in all of computer science is how we are to get to the situation where we build on top of the work of others rather than redoing so much of it in a trivially different way. Science is supposed to be cumulative, not almost endless duplication of the same kind of things.
>
> *Richard Hamming*
> *1968 Turning Award Lecture*

In contemporary's digital world, organizations face unprecedented challenges in complex and dynamic market environments, such as volatile customer demands, increasing market dynamics, and the continuous emergence of new advancements in information technology [1, 2, 3, 4]. Software development projects in such environments face changes either directly or indirectly [5]. These changes also affect approaches that are used for software development [6]. Traditional software development approaches are considered insufficient for addressing organizational needs as they are not designed to detect relevant changes and respond in a timely and effective manner [7]. Consequently, a movement sparked by various software development practitioners emerged in the 1990s with new software development approaches to address the aforementioned hurdles, culminating in the creation of the Agile Manifesto [8] and many agile software development methods [9], including eXtreme Programming (XP) [10], Kanban [11], and Scrum [12].

The initial application of these methods has targeted small, co-located, and self-organizing teams that develop software in close collaboration with business customers, relying on regular feedback and rapid development iterations [13]. The successful implementation of agile methods in small projects has inspired organizations to apply them outside of their sweet spot in complex and large projects [14]. Thus, the popularity of using agile methods in large-scale projects has increased in both practice and academia over the past decade [15, 16]. This trend is also confirmed by Uludağ et al. [17] stating that 47.79% of all related research on large-scale agile development was published in 2018 and 2019, making the relevance of this research field even more important than ever. From a practical perspective, adopting agile methods at scale also entails unprecedented challenges, such as inter-team coordination and communication [18]. Thus, large organizations are engrossed in enhancing agile methods to incorporate larger teams and tackle related challenges [18]. From a scientific perspective, the literature about large-scale agile development is still maturing compared to the literature regarding agile software development having a profound body of knowledge [17]. Similar to challenges and best practices in agile software development [19], researchers aim to identify challenges and best practices regarding the adoption of agile methods at large-scale [20]. However, solely documenting and identifying challenges and patterns in large-scale agile development is insufficient as they have to be presented to their adopters understandably and appropriately [21]. Against this backdrop, this master's thesis's main objective is to identify requirements for establishing online pattern communities in large-scale agile development and implement the identified requirements by extending a prototypical web implementation.

1.2 Research Objectives

Resulting from the motivation in Section 1, the main objective of this master's is to investigate the challenges of establishing pattern communities and support the establishment of such communities. Based on this objective, three research questions (RQs) were formulated.

> ***Research question 1:*** *What are the challenges of establishing pattern communities?*

The first research question's objective is to identify literature on large-scale agile development, pattern communities, and the interplay between these areas with the help of a literature review.

> ***Research question 2:*** *How can a prototypical web implementation support the establishment of pattern communities?*

The second research question's objective is to map derived challenges for building pattern communities as requirements and features for the web implementation. Since the prototypical web implementation aims to create a platform that enables sociability and interactivity, this research question aims to identify and implement the social design guidelines recommended by Kraut and Resnick [22].

> ***Research question 3:*** *How can the prototypical web implementation be improved in the future?*

The third research question aims to compare the prototypical web implementation with existing pattern communities and literature to provide suggestions for future work.

1.3 Research Approach

The following section gives an overview of the overall research methodology applied in this thesis.

In this thesis, design science research was chosen as a research approach to answer the RQs defined in Section 1.2. The design science research approach was introduced by Hevner et al. [23] and further developed by Peffers et al. [24]. The design science research approach is an adequate research methodology to answer the RQs of this thesis as it centers around generating and evaluating new and innovative artifacts inward a problem domain, intending to solve identified problems. We aim to find the right balance between ensuring practical connection and applicability. Design science research provides the connection, whereas the literature review and related work contribute to the necessary rigor of the conducted research.

As shown in Figure 1.1, the design science approach consists of six phases explained in the following. In the first phase, the existing literature is reviewed to understand the state of the problem. Goals and requirements are formulated at the end of this phase. In the second phase, the research questions and the objectives of the potential solution are defined. Also, in this phase, requirements for the prototype are elicited. In the third phase, the prototype to address the requirements is designed and developed. In the fourth phase, the prototype is demonstrated to its potential users to provide proof of concept and show that the designed prototype is consistent with successfully undertaken design science research. In the fifth phase, the analytic data of the platform is assessed to evaluate the platform. Finally, in the sixth phase, the results of this research are being published in this thesis.

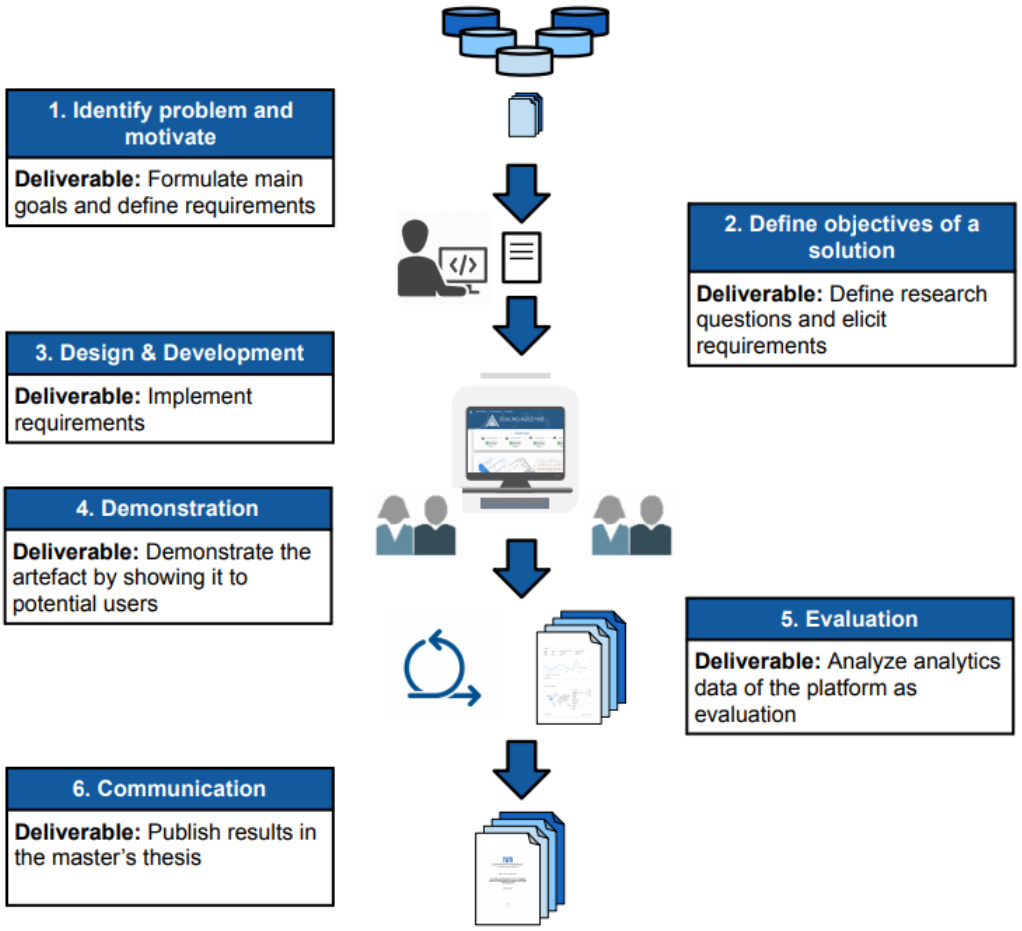


Figure 1.1: Overview of the research approach [24]

The remainder of this thesis is structured as follows. In Chapter 2, fundamental terminologies, such as agile and large-scale agile development, are defined and described. In Chapter 3, an overview of relevant related work is presented. In Chapter 4, the solution artifact, namely the prototypical web implementation is presented. In Chapter 5, the evaluation of the prototypical web implementation is demonstrated by providing an overview of related analytics data. In Chapter 6, the key findings and limitations of this thesis are presented before concluding the thesis with a summary of results and remarks on future research in Chapter 7.

2 Foundations

This chapter is dedicated to describing the theoretical foundations of this thesis. First, in Section 2.1, the ideas behind agile software development are presented. Based on this, in Section 2.2, the term large-scale agile development is further explored. Finally, a brief outline on social design aspects that are used later in the thesis is given in Section 2.3.

2.1 Agile Software Development

This section aims to create a unified view of agile software development. First, the Agile Manifesto is introduced by presenting the values and principles behind it. Afterward, two of the most popular agile and lean software development methods, namely Scrum and Kanban, are presented.

2.1.1 Agile Manifesto

In 2001, seventeen professionals formulated the manifesto to identify and describe an alternative approach to traditional software development processes by defining values and basic principles for better software development [25]. The word "*agile*" was chosen to construe the attributes of this different approach [26]. All the participants signed the Agile Manifesto to indicate the consensus on the principles and values it contains [27]. The manifesto is commonly introduced whenever developers aim at developing software based on the following values and principles [19]:

> "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> - **Individuals and interactions** over processes and tools
> - **Working software** over comprehensive documentation
> - **Customer collaboration** over contract negotiation
> - **Responding to change** o over following a plan
>
> That is, while there is value in the items on the right, we value the items on the left more."

Principle	Quotation
P1	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
P2	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
P3	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
P4	Business people and developers must work together daily throughout the project.
P5	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
P6	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
P7	Working software is the primary measure of progress.
P8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
P9	Continuous attention to technical excellence and good design enhances agility.
P10	Simplicity–the art of maximizing the amount of work not done–is essential.
P11	The best architectures, requirements, and designs emerge from self-organizing teams.
P12	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Table 2.1: Principles behind the Agile Manifesto by Beck et al. [8]

In a virtual workshop on agile methodologies organized by the Centre of Experimental Software Engineering, the participants specified agile methodologies as iterative, incremental, self-organizing, and emergent. In addition, they declared that all agile methodologies follow the four values and twelve principles of the Agile Manifesto (see Table 2.1) [28]. In the following, the four values are extended by an interpretation by Stellman and Greene [29]:

- "Individuals and interactions over processes and tools" means that it is more important for agile teams to focus on the team members and their communication than to pay attention to the tools and practices used.
- "Working software over comprehensive documentation" states that software which satisfies a user's need is more valuable than a specification describing the need.
- "Customer collaboration over contract negotiation" means that from now on, customers will be seen as team members.

- "Responding to change over following a plan" means that plans rarely come true and that it is, therefore, more important to deliver software than to work on a plan.

While agile methodologies provide benefits over traditional software development methodologies, such as waterfall, in terms of responding to changes in objectives, resources, materials, techniques, and tools, a remarkable amount of work is usually done before the project starts to use the agile methodologies effectively [30, 31].

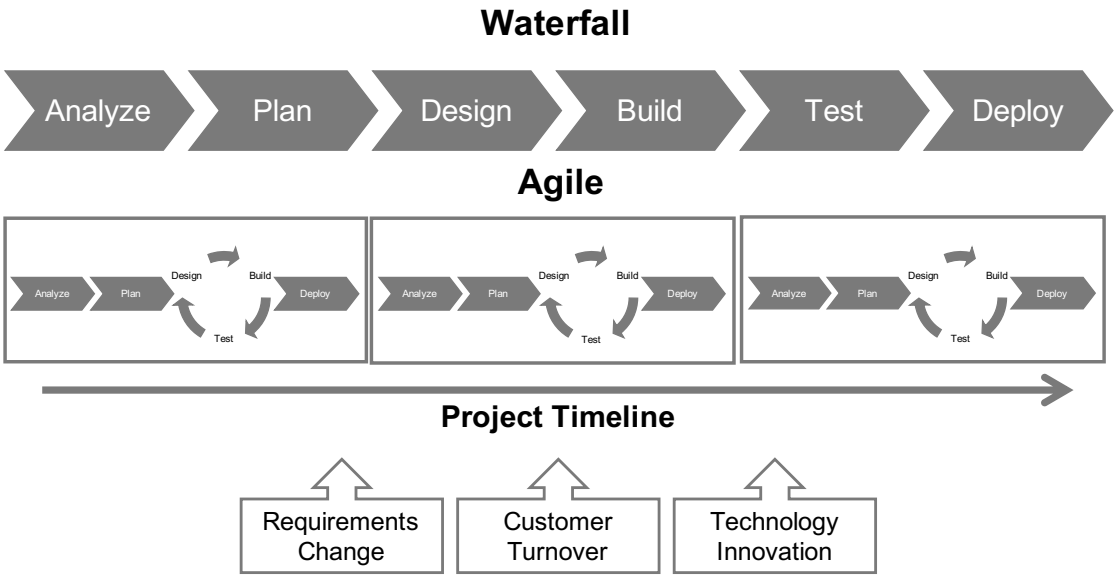


Figure 2.1: Design phase composition between waterfall and agile development [32]

2.1.2 Scrum Process Framework

> One of the charms of the Rugby Union game is the infinite variety of its possible tactics. [...] with the ball in its hands, a team is in a position to dictate tactics which will make the best use of its own particular talents, at the same time probing for and exposing weaknesses in the opposing team. The ideal team [...] will make sure that the possession won by the forwards is employed to the maximum embarrassment of the opposing team.
>
> *Takeuchi and Nonaka*
> *The new product development game [33]*

Scrum is defined as a framework within which people can address complex adaptive

problems while productively and creatively delivering products of the highest possible value [34]. The framework was named after the Scrum formed in rugby sports [35]. Ken Schwaber and Jeff Sutherland are the co-creators of the Scrum process framework inspired by the revolutionary work of Takeuchi and Nonaka [33]. Figure 2.2 provides an overview of Scrum. The three essential components of the Scrum framework are roles, process, and artifacts [36]. These components are described in detail below.

Roles

In Scrum, the project duties are delegated to the three roles: Developer, Product Owner, and Scrum Master [38]. An essential characteristic of Scrum is that each team member can individually determine how to address the tasks assigned to ensure that the Scrum mechanism can not be interfered with by anyone [38, 39]. In direct and consistent collaboration with clients, users, and other stakeholders, the Product Owner of a Scrum project identifies the product specifications and then communicates them to the team that implements the requirements [40]. The Product Owner is also responsible for the project's success and therefore for ensuring that the product satisfies the expectations of the consumer [40]. The Scrum team's main objective is to fulfill the specifications identified by the Product Owner[38]. Scrum teams share the following aspects: they are cross-functional, self-organizing, and consist of five to ten participants [36]. The Scrum Master's role is to build the Scrum practices and values and eliminate obstacles to make a smooth Scrum process [36, 40]. The Scrum Master fulfills a support role for the process and is responsible for its effectiveness [41].

Artifacts

The three core components included in Scrum are the Product Backlog, Sprint Backlog, and Burndown Charts. The agile project specifications are documented, allocated, and prioritized using both the Product Backlog and the Sprint Backlog [38]. The third artifact, the Burndown Chart, is a method that enables the prediction of how much time a project takes until the product satisfies all consumer needs, depending on the specifications [38].

The Product Backlog is a list of all project specifications that is continually changing as it adapts to the modifications faced by the project [38]. The Product Backlog is no longer be used if the project lacks funds [38]. The Product Owner is responsible for ensuring that the Product Backlog is available and ensuring the consistency of all its submissions [38]. The Product Backlog can be modified at the time of the project kickoff and concurrent Sprint preparation sessions, recurring in line with the project cycle, [36].

Backlogs can be interpreted as a collection of requirements. The Sprint Backlog is the

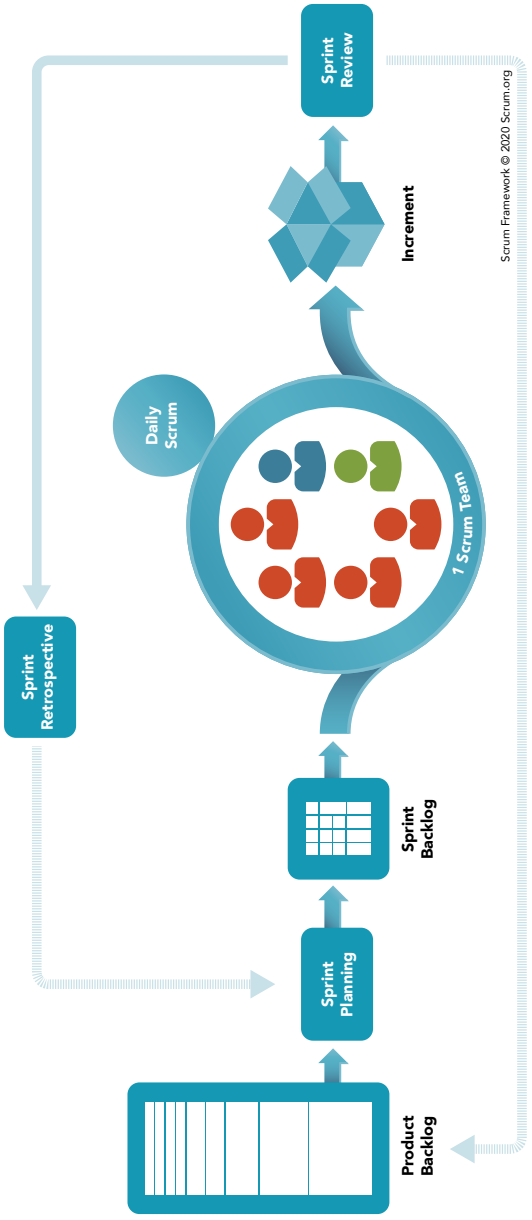


Figure 2.2: Overview of the Scrum Framework [37]

subset of the Product Backlog that includes exactly as many requirements as the team can accomplish in one Sprint by the team [36]. The fact that the number of requirements correlates to the team's estimated capacities is an essential feature of Scrum, which allows only the team to determine the requirements moved to the Sprint Backlog from the Product Backlog [36, 40].

The Burndown Chart is the last artifact in Scrum, which represents the time left before the project finishes [38]. An example project Burndown Chart is shown in Figure 2.3. In this example, remaining story points are illustrated over time.

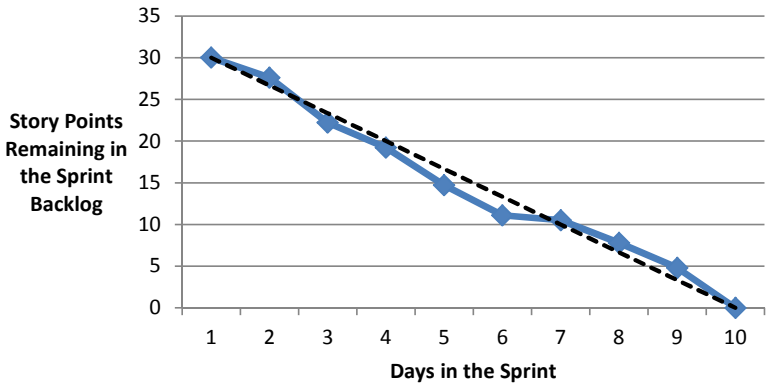


Figure 2.3: Burn Down Chart [42]

Process

Scrum embraces a concept in which all its activities are centered to include products with the ultimate possible value [38]. Every Scrum project has a vision about the system built through the project [38]. First, the Product Owner must draw from this vision all the required criteria for the execution of the vision. Second, the Product Owner must ensure that the implementation is structured to optimize the return on investment of the project's funding investors [38]. The iterative and incremental mechanism used to implement the Product Backlog specifications is the fundamental building block of Scrum [38]. Every iteration starts after the Sprint Planning. In tight collaboration with the Product Owner and the team members, the Product Backlog's highest priority criteria are defined and passed to the Sprint Backlog [38]. The team works on the Sprint Backlog items during a Sprint and gathers every day to hold a Daily Scrum to synchronize the team members' work [38]. The Scrum Master invites all team members to participate in an event called the Scrum Retrospective held after the Sprint review meeting. The Scrum Retrospective is the last event of the iteration and seeks to make the team focus on the previous Sprint and determine steps that allows for a successful

and pleasant next Sprint [38]. The Scrum retrospective continues with the next Sprint planning meeting as it starts the next iteration [38].

2.1.3 Kanban in Software Development

Kanban was first applied successfully at Toyota, a corporation operating in the Japanese automotive industry, in the 1950s [43, 44]. Sixty-four years later, Kanban was brought by David J. Anderson into the software development business at Microsoft, who led an under-performing software team back to a promising path with Kanban [43, 44]. Although Kanban has been important in the software industry, the first attempts have been made to merge Kanban with Scrum, a well-established agile software development methodology [45]. While Scrum practices incremental and iterative methods to create products of the maximum possible value, Kanban focuses on enhancing the workflow, minimizing a work item's total production time, and restricting work in progress [46, 47]. Table 2.2 shows the benefits of applying Kanban.

Number	Benefit
N1	Better understanding of whole processes
N2	Improved software quality
N3	Improved meeting of customer needs and customer satisfaction
N4	Increased motivation of engineers
N5	Improved communication/coordination between stakeholders/ in team
N6	Bugs were fixed more quickly, work in progress made it easier to handle blocking in work
N7	Increased software productivity
N8	Problem solving (easy detection and removal of bugs)
N9	Reduced batch size
N10	Decreased time to delivery
N11	Increased release frequency
N12	Efficiently controlled software projects
N13	Changes to requirements made welcome
N14	Early feedback on features, without delays
N15	No massive documents (limited to customer request)
N16	Task approval from management not needed – approval gotten from customer in demos

Table 2.2: Benefits of applying Kanban [44]

In the literature, a combination composed of Kanban and Scrum elements is also

called Scrumban [**ladas2009scrumban nikitina2012scrum**, 48, 49, 50]. The purpose of Scrumban is to help enterprises who are continuously subjected to evolving consumer needs and recurrent coding issues [48]. Kanban is a depiction of a pull mechanism to manage the workflow used in software development projects [51]. Workflow management is achieved by setting a work in progress threshold, defining constraints, and organizing team activity [51]. The work in progress limit is set based on the work capacity of the team. This allows a balance to be established between the demand and results [44]. This balance encourages sustainable growth, which can, on the one hand, contribute to increased team success and, on the other hand, increase the consistency of the products developed [44]. Kanban board, which is the visual representation of a value stream, helps implement the Kanban in software projects [52]. This value stream is split into columns reflecting the work's current status and the work objects flow [52]. However, only a predefined number of items is permitted in any of these columns, which means that the work in progress is constrained [52]. As soon as predefined conditions are fulfilled, an item leaves one column and transfers to the next [52]. Finally, a value stream can be noticed by viewing objects' roaming through their working states [52].

2.2 Large-Scale Agile Development

> My dear, here we must run as fast as we can, just to stay in place. And if you wish to go anywhere you must run twice as fast as that.
>
> *Lewis Carroll*
> *Alice in Wonderland*

The effectiveness of agile methodologies in small, co-located teams and their widespread use promoted their usages in new domains [53, 54]. Dingsøyr et al. [53, 54] defines the term "*large-scale agile development*" as a development effort with "more than two teams" and "*very large-scale agile development*" efforts with "more than ten teams". Also, a distinction can be made between "*large-scale agile development*" and "*enterprise agile*" which refers to the implementation of agile development methodologies, principles, and values for the whole organization and not just software development activities [54]. A case study revealed that agile practices' implementation leads to increased recognition of initiatives, increased exchange of information, and stronger collaboration on a large scale [55]. On the other hand, when attempting to scale agile techniques, there are many documented drawbacks and obstacles. An analysis of three large-scale agile development cases illustrated serious problems, especially the lack of guidance on agile approaches related to team-to-team dependencies and inter-team coordination [56]. It

takes a lot of management effort to get all the agile teams collaborating towards a shared goal [57]. In large-scale applications, information sharing is quite significant since expertise might be distributed to different locations and teams [58]. Agile approaches are highly troublesome in large-scale system architectures and applications when integrating current and emerging software architectures [59]. According to Leffingwell and his peers, some architectural planning and governance are necessary to produce and maintain such systems reliably. Individuals teams, products, and programs may not even have the visibility necessary to see how the larger enterprise system needs to evolve [60].

2.2.1 Scaling Agile Frameworks

Based on a structured literature review, Uludağ et al. [5] identified twenty scaling agile frameworks (see Table 2.3). Most frameworks are based on traditional agile approaches such as XP and Scrum and have been tailored to modern standards for large projects in which multiple teams operate. Also, Uludağ et al. [5] calculated maturity scores for the frameworks. Large Scale Scrum, Scaled Agile Framework, and Disciplined Agile 2.0 are the three frameworks with the best maturity ratings.

Scaling agile framework	Author	Organization	Publication year
Crystal Family	Alistair Cockburn	N/A	1992
Dynamic Systems Development Method Agile Project Framework for Scrum	Arie van Bennekum	DSDM Consortium	1994
Scrum-of-Scrums	Jeff Sutherland and Ken Schwaber	Scrum Inc.	2001
Enterprise Scrum	Mike Beedle	Enterprise Scrum Inc.	2002
Agile Software Solution Framework	Asif Qumer and Brian Henderson-Sellers	University of Technology	2007
Large Scale Scrum	Craig Larman and Bas Vodde	LeSS Company B.V.	2008
Scaled Agile Framework	Dean Leffingwell	Scaled Agile Inc.	2011
Disciplined Agile 2.0	Scrott Ambler	Disciplined Agile Consortium	2012
Spotify Model	Henrik Kniberg, Anders Ivarsson, and Joakim Sundén	Spotify	2012
Mega Framework	Rafael Maranzato, Marden Neubert, and Paula Heculano	Universo Online S.A	2012
Enterprise Agile Delivery and Agile Governance Practice	Erik Marks	AgilePath	2012
Recipes for Agile Governance in the Enterprise	Kevin Thompson	Cprime	2013
Continuous Agile Framework	Andy Singleton	Maxos LLC	2014
Scrum at Scale	Jeff Sutherland and Alex Brow	Scrum Inc.	2014
Enterprise Transition Framework	N/A	agile42	2014
ScALeD Agile Lean Development	Peter Beck, Markus Gärtner, Christoph Mathis, Stefan Roock and Andreas Schliep	N/A	2014
eXponential Simple Continuous Autonomous Learning Ecosystem	Peter Merel	Xscale Alliance	2014
Lean Enterprise Agile Framework	N/A	LeanPitch Technologies	2015
Nexus	Ken Schwaber	Scrum.org	2015
FAST Agile	Ron Quartel	Cron Technologies	2015

Table 2.3: Overview of scaling agile frameworks [5]

2.2.2 Large-Scale Agile Development Patterns

The implementation of agile approaches on a large scale raises particular problems and challenges [61] such as coordination complexity, difficult architectural integration and increased stakeholder numbers [62]. Addressing these challenges is the path to reach the maximum advantages of agility in large-scale environments [63]. The identification of recurring concerns and documentation of best practices gives the impression to be effective [64]. Thus, Uludağ et al. [20], introduced a conceptual pattern language for large-scale agile development which forms the basis for documenting of recurring concerns and patterns (see Figure 2.4).

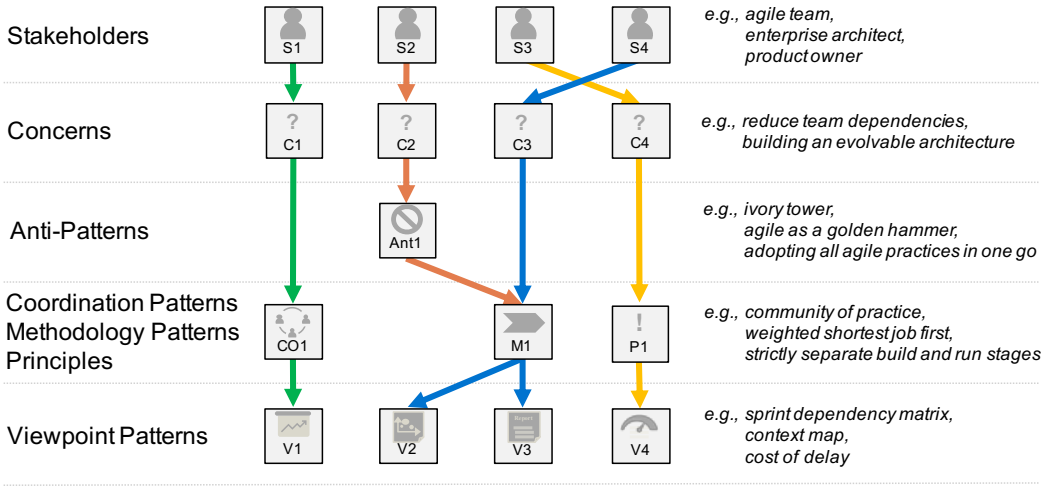


Figure 2.4: Conceptual overview of the Large-scale Agile Development Pattern Language
[20]

The pattern language consists of three types of patterns and four concepts [20]:

- **Stakeholders** are all persons who are actively involved in, have an interest in, or are in some way affected by large-scale agile development.
- **Concerns** can manifest themselves in many forms, e.g., expectations, goals, needs or responsibilities.
- **Principles** are general rules and guidelines that address given concerns by providing a common direction for action. In comparison to patterns, they do not provide any descriptions on 'how' to address concerns.

- **Coordination Patterns (CO-Patterns)** document proven coordination mechanisms to address recurring coordination concerns, i.e., managing dependencies between activities, resources or tasks.
- **Methodology Patterns (M-Patterns)** document concrete steps to be taken to address given concerns.
- **Viewpoint Patterns (V-Patterns)** document proven ways to visualize information in form of boards, documents, metrics, models, and reports to address recurring concerns.
- **Anti-Patterns** document typical mistakes and present revised solutions, which help pattern users to prevent these pitfalls.

Figure 2.5 depicts the conceptual model and the core elements used to document the concepts and patterns of the pattern language.

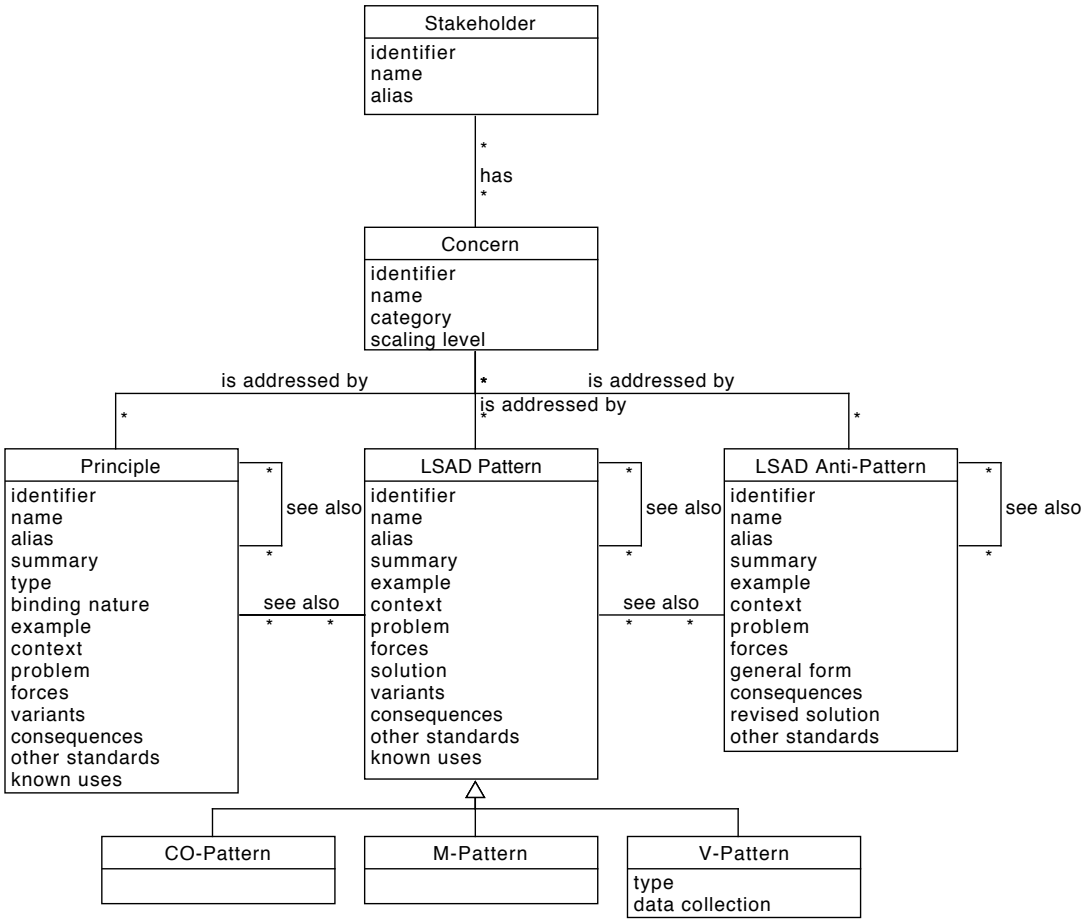


Figure 2.5: Conceptual model of the Large-Scale Agile Development Pattern Language [20]

2.3 Social Design Principles

Based on social science and psychology studies, desirable behaviors can be promoted through certain design concepts in communities [22]. Kraut and Resnick [22] listed design claims in the chapter "*Encouraging Contribution to Online Communities*". Not only are these design concepts applicable to an online community, but many of them come from research in the social psychology experiments and general social science findings, which also make them important to a community approach in general. In this thesis, the detailed design principles with relevance and implementation are described in detail in Chapter 4.

3 Related Work

This chapter summarizes primary publications that are relevant for this thesis in the areas of large-scale agile software development and software pattern communities. The presented related work further extends and elaborates on the foundations presented in the last chapter.

Henninger et al. (2007)

Henninger et al. [21] performed initial research to reveal problems associated with the use of patterns among a set of pattern collections. Henninger et al. [21] identified six challenges through the empirical work to federate the currently separate realm of pattern collections into a more integrated and interconnected body of knowledge to build online pattern communities. The identified challenges are strongly biased towards uniting heterogeneous patterns using web technologies in a distributed electronic format.

Challenges for federating software patterns:

- Electronic accessibility
- Lack of standard pattern forms
- Inter-pattern relationships
- Software pattern validation
- Tracking software pattern variants and duplicates
- Updating software pattern knowledge

Dikert et al. (2016)

A systematic literature review was conducted by Dikert et al. [61] to reveal challenges and factors that are influencing the success of at large-scale agile transformations. Thus, 52 papers describing 42 different organizations were analyzed. The authors stated that the identified success factors and challenges are significantly important. Management

support, communication, and transparency, mindset, and alignment, as well as team autonomy, are some of the success factors that were identified by the literature review. On the other hand, resistance to change and the coordination of multiple teams in a large-scale environment are a few of the identified challenges.

Uludağ et al. (2018)

Uludağ et al. [65] aimed to identify typical challenges of stakeholders in the area of large-scale agile development by analyzing 73 relevant sources. As a consequence, 79 challenges were reported by 14 distinguished stakeholders and divided into 11 categories.

Stakeholders:

- Agile coach
- Business analyst
- Development team
- Enterprise architect
- Portfolio manager
- Product manager
- Product owner
- Program manager
- Scrum master
- Software architect
- Solution architect
- Support engineer
- Test team
- UX expert

Categories:

- Culture and mindset
- Communication and coordination
- Enterprise architecture
- Geographical distribution
- Knowledge management
- Methodology
- Project management
- Quality assurance
- Requirements engineering
- Software architecture
- Tooling

The research by Uludağ et al. [65] revealed a lack of literature addressing challenges and best practices in large-scale agile development. Thus, the Large-Scale Agile Development Pattern Language was created based on this research by Uludag et al. [20].

Uludağ et al. (2019)

Agile coaches and scrum master are confronted with a number of unprecedented concerns in large-scale agile development. Uludağ et al. [66] conducted 13 interviews with agile coaches and scrum masters to identify the concerns and best practices. As a result, 57 recurring concerns and 15 best practices were identified.

Uludağ and Matthes (2020)

Based on 13 expert interviews and 45 case study interviews, Uludağ and Matthes [67] identified a total of 43 patterns for addressing recurring concerns of enterprise and solution architects. In addition, they revealed 35 recurring concerns, of which 16 have already been identified in the structured literature review.

ScrumPLoP

All the presented patterns at ScrumPLoP conferences [68] are assembled on the ScrumPLoP website which currently contains 234 patterns. James Coplien, Neil Harrison, and Mike Beedle are important authors who contributed to the collection. Each pattern includes a picture, context, force, 'therefore', examples, related patterns, and references. Since there is no guiding structure, the identification of related patterns is difficult. Some of the patterns are also related to large-scale agile development.

4 Implementation

This section describes the prototypical implementation of a web application aiming to support the establishment of pattern communities at large-scale agile development. Its features are designed to solve the observed problems of pattern communities [21], such as:

- Electronic accessibility
- Lack of standard pattern forms
- Pattern validation
- Updating pattern knowledge
- Lack of feedback

The aim is to enhance the prototypical web application with certain social design concepts that promote desirable behaviors. The main focus is making the prototypical web application interactive while addressing the identified challenges.

In the beginning of this chapter, Section 4.1 explains the motivation and rationale for developing a web application. Subsequently, Section 4.2 briefly outlines the technical requirements and technology selection. Section 4.3 shows the system architecture. Finally, the main views and core features are presented in Section 4.4.

4.1 Motivation for a Web Application

To tackle the described challenges and achieve the solution goals and improvements, using a web application is only one possible answer. While there might be other solutions to address the identified challenges, the decision for a web application made because of the following arguments:

- Scalability, because a web application can better support larger communities.
- Accessibility and portability to access the information on patterns from any location and different devices.

- The interactivity and possible implementation and usage of social design principles to encourage contribution and participation and other desirable behavior.
- The possibility of a web application to communicate with other tools, development platforms, and databases for data analysis, e.g., to improve the tool by analyzing collected data.
- The ease of adding and updating patterns so that new patterns can be easily added to the platform.

4.2 Technical Requirements, Technology selection, and Usage

The technical requirements for the prototype are straightforward. The application can be easily deployed to the chair's infrastructure. Also, it should use the technologies that are suggested by the chair which are SocioCortex[1] for the backend system, React[2] framework for the frontend and NodeJS[3] for external functionalities. As the UI component library, material-ui[4] and for the version control system, Git[5] is used. To manage Git repositories in a better way, GitHub[6] is selected which is a cloud-based hosting service.

[1] `https://sociocortex.com/`, last accessed on: 04-09-2021.
[2] `https://reactjs.org/`, last accessed on: 04-09-2021.
[3] `https://nodejs.org/`, last accessed on: 04-09-2021.
[4] `https://material-ui.com/`, last accessed on: 04-09-2021.
[5] `https://git-scm.com/`, last accessed on: 04-09-2021.
[6] `https://github.com/`, last accessed on: 04-09-2021.

4.2.1 SocioCortex

This prototype uses SocioCortex as the technical environment. SocioCortex is an information system to organize semi-structured data within Enterprise Architecture Management, employing a dynamic and collaborative Wiki-based approach developed by the chair of Software Engineering for Business Information Systems (SEBIS) of the Technische Universität München [69]. It is based on the modeling framework of a former tool called Tricia [70]. The data model is depicted in Figure 4.1. Data is structured and presented as interconnected Wiki pages. While SocioCortex implements Wiki pages features, it also brings a standardized REST-API to enable access to all features. Furthermore, the platform provides a Model-based Expression Language (MxL) which is a powerful query language to access the data in the system. Data in the SocioCortex system has to be queried to fill in the parameters with values. The Model-Based Language expression is used for this purpose. MxL is a domain-specific language developed on SocioCortex's data model. The following characteristics characterize MxL [71]:

1. **Functional**
   The language is characterized by invoking functions. As a consequence of a typical query operation, like "select", a corresponding function gets called (select-function, where-function, etc.)

2. **Sequence oriented**
   MxL concentrates on the usage of sequences (ordered sets) and supplies various functions to support these.

3. **Object oriented**
   SocioCortex entities are considered as objects and entityTypes as classes. Therefore data-model can be queried.

4. **Statically type safe**
   The static semantics is validated as soon as the user enters a query. By analyzing semantic dependencies, automated refactoring is possible.

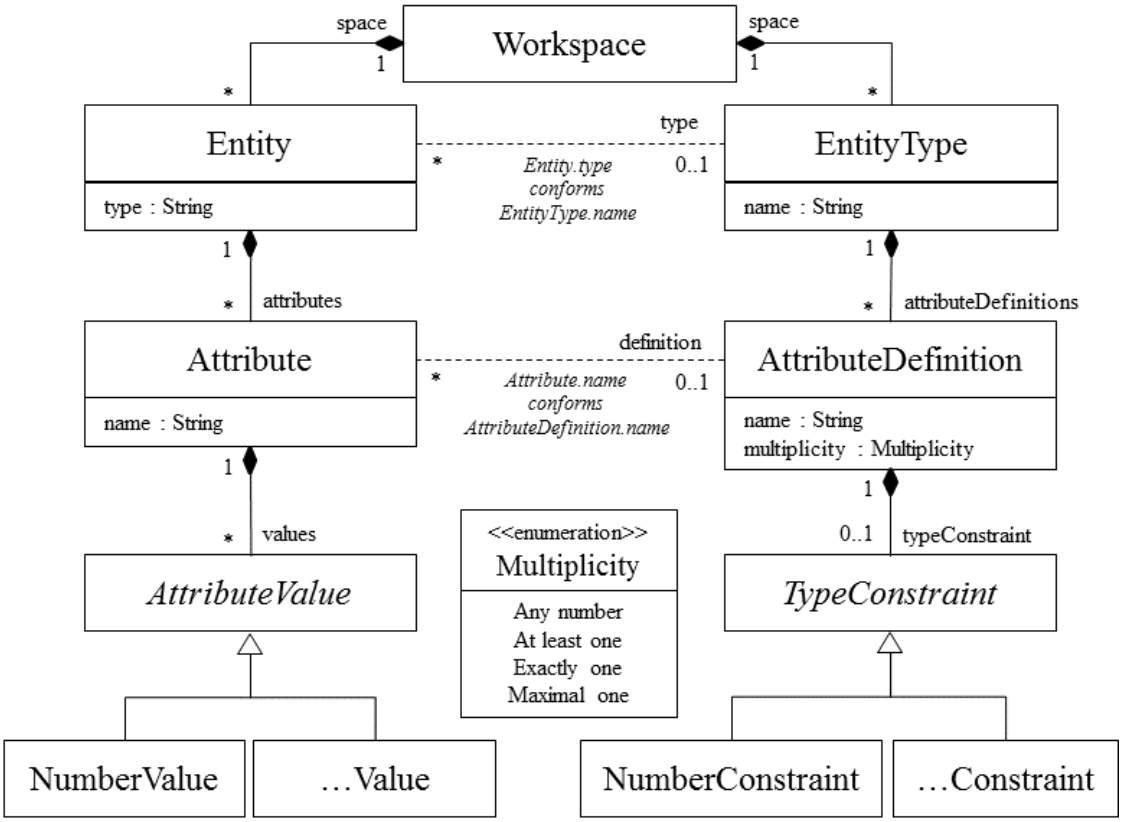


Figure 4.1: Hybrid-Wiki data model in the context of Enterprise Architecture Management [71]

Simple and complex attribute types are part of the type-system of MxL. The types depicted in Figure 4.2 are supported.

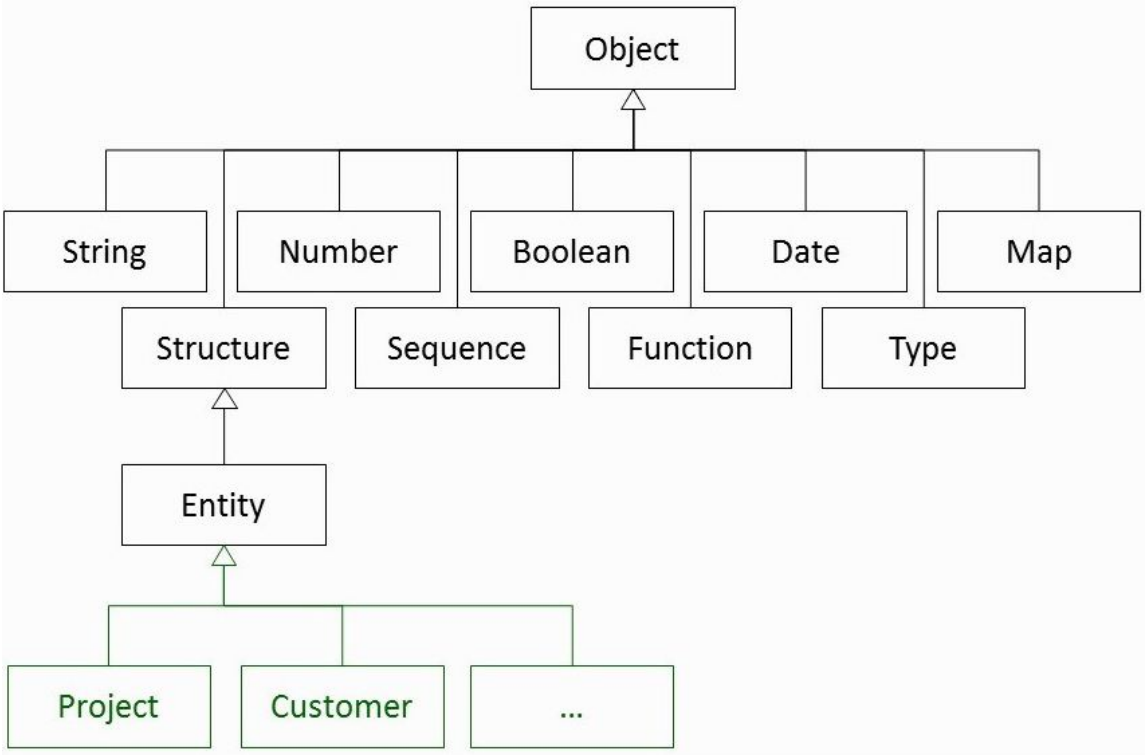


Figure 4.2: Overview of all basic types of MxL in form of an UML class diagram [72]

Accessing SocioCortex

MxL is used to query a SocioCortex data model. But also, a function of creating and editing entities is necessary. Thus, SocioCortex also serves as a repository to store the created artifacts. Furthermore, an authentication mechanism also is needed. A REST-API is provided for this purpose by SocioCortex. Listing 4.1 and Listing 4.2 show most crucial operations for the application provided by SocioCortex.

```
static sendPost(url, body, headers) {
  const fetchData = {
    method: 'POST',
    ...body && { body: JSON.stringify(body) },
    ...headers && { headers: headers },
  }

  return fetch(url, fetchData)
    .then(response => {
      if (!response.ok) {
        throw Error(response.statusText)
      }
      return response
    })
    .then(response => response.json())
    .then(data => data)
}

static sendPut(url, body, headers) {
  const fetchData = {
    method: 'PUT',
    ...body && { body: JSON.stringify(body) },
    ...headers && { headers: headers },
  }

  return fetch(url, fetchData)
    .then(response => {
      if (!response.ok) {
        throw Error(response.statusText)
      }
      return response
    })
    .then(response => response.json())
    .then(data => data)
}
```

Listing 4.1: POST and PUT requests of the SocioCortex REST-API

```
static sendGet(url, headers) {
  const fetchData = {
    method: 'GET',
    ...headers && { headers: headers },
  }

  return fetch(url, fetchData)
    .then(response => {
      if (!response.ok) {
        throw Error(response.statusText)
      }
      return response
    })
    .then(response => response.json())
    .then(data => data)
}

static sendDelete(url, headers) {
  const fetchData = {
    method: 'DELETE',
    ...headers && { headers: headers },
  }

  return fetch(url, fetchData)
    .then(response => {
      if (!response.ok) {
        throw Error(response.statusText)
      }
      return response
    })
    .then(response => response.json())
    .then(data => data)
}
```

Listing 4.2: GET and DELETE requests of the SocioCortex REST-API

Furthermore, example usage of SocioCortex's REST-API is illustrated in Listing 4.3.

```
static getFeedbackQuery() {
  return (
    "find('Feedback').select({" +
    "id: id," +
    "name: Name," +
    "comment: comment," +
    "isSubCommentOf: isSubCommentOf," +
    "pattern: pattern," +
    "patternName: patternName," +
    "star: star," +
    "timestamp: timestamp," +
    "upvotes: upvotes," +
    "userid: userid," +
    "username: username" +
    "})"
  );
}
static getFeedback() {
  return {
    promise: FetchService.sendPost(FEEDBACK_MXL_URL, {
      expression: this.getFeedbackQuery(),
    }),
  };
}
```

Listing 4.3: Example implementation of SocioCortex REST-API

4.2.2 React

React is an open-source library developed by Facebook to implement user interfaces [73]. React facilitates the development of single-page applications. The core of React are components and their compositions which are being realized as JSX files. According to Chinnathambi [74], visuals can be defined in JSX files with a syntax similar to HTML but still getting the power and flexibility from JavaScript. Chinnathambi [74] introduces React components as reusable chunks of JavaScript that output (via JSX) HTML elements. They contain both the control and the view. Since component logic is written in JavaScript instead of templates, it is easy to pass rich data through the app and keep the state out of the Document Object Model (DOM). React components implement a render method that takes input data and returns what to display. An example is shown in Listing 4.4, which uses an XML-like syntax called JSX.

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  document.getElementById('hello-example')
);
```

Listing 4.4: An example React component

When views are designed for each state in the application, React will efficiently update and render just the right components when the data changes. Declarative views make the code more predictable and easier to debug. For the application, encapsulated components are built, which manage their state, and then those components are composed to make complex user interfaces. Example usage is shown in Listing 4.5.

```
import React from "react";

import HomeViewComponent from "../components/Home/HomeViewComponent";
import Header from "../components/Header";
import Footer from "../components/Footer";
import CircularProgress from "@material-ui/core/CircularProgress";

const HomeView = (props) => {
  const { history } = props;
  const [allData, setAllData] = React.useState();
  const [isLoading, setIsLoading] = React.useState(true);

  const handleSetData = (data) => {
    setAllData(data);
  };

  if (isLoading) {
    return (
      <CircularProgress size={75} thickness={4} />
    );
  }

  return (
    <div>
      <Header history={history}/>
      <HomeViewComponent
        allData={allData}
      />
      <Footer />
    </div>
  );
};

export default HomeView;
```

Listing 4.5: Example usage of multiple components

Furthermore, the firebase-analytics[7] package is used to keep track of analytics data for the application. For this reason firebase.js is implemented which can be seen in Listing 4.6.

```
import firebase from "firebase";
import "firebase/analytics";

const firebaseConfig = {
  apiKey: process.env.API_KEY,
  authDomain: process.env.AUTH_DOMAIN,
  databaseURL: process.env.DATABASE_URL,
  projectId: process.env.PROJECT_ID,
  storageBucket: process.env.STORAGE_BUCKET,
  messagingSenderId: process.env.MESSAGING_SENDER_ID,
  appId: process.env.APP_ID,
  measurementId: process.env.MESASUREMENT_ID,
};

// Check that `window` is in scope for the analytics module!
if (typeof window !== "undefined" && !firebase.apps.length) {
  firebase.initializeApp(firebaseConfig);
  if ("measurementId" in firebaseConfig) firebase.analytics();
}

export default firebase;
```

Listing 4.6: firebase.js file for the application

[7] `https://github.com/firebase/firebase-js-sdk`, last accessed on: 04-09-2021.

The file is imported into app.js file. With this implementation, it is possible to see default analytics data reported by the frontend application. However, it is aimed to keep track of some specific events, such as pattern catalog downloads. To achieve this, firebase-analytics package logEvent functionality is used. Example implementation for logging events can be seen in Listing 4.7.

```
onClick={() => {
  if (firebase.apps.length) {
    firebase.app(); // if already initialized, use that one
  }
  if ("measurementId" in firebaseConfig) {
    firebase.analytics();
    firebase.analytics().logEvent("full_catalog");
    console.log("logged");
  }
```

Listing 4.7: Example usage of the logEvent function

Moreover, Figure 4.3 shows the overall folder structure for the React application.

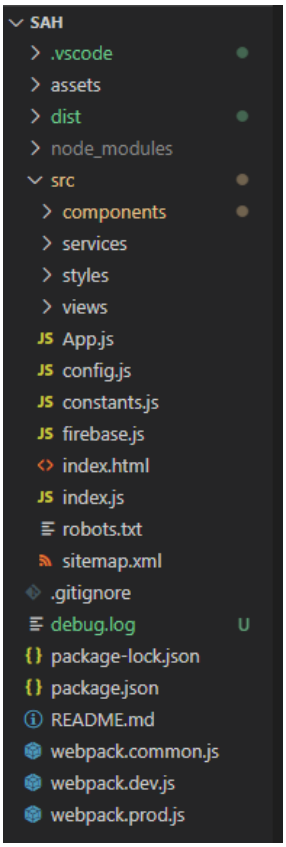


Figure 4.3: Overall structure for the React application

4.2.3 NodeJS

NodeJS, an asynchronous event-driven JavaScript runtime, is designed to build scalable network applications [75]. It enables running JavaScript code on any machine where NodeJS can be installed. Thread-based networking is relatively inefficient and very difficult to use [75]. NodeJS is single-threaded, which means that it executes one line of code at a time. Satheesh et al. [76] describe the goal that NodeJS tries to solve as follows: "It tries to do asynchronous processing on a single thread to provide more performance and scalability for applications that are supposed to handle too much web traffic." Thus, it is asynchronous, which means it does not execute the lines of code chronologically from top to bottom but can do multiple operations simultaneously. Furthermore, users of NodeJS are free from the worries of dead-locking the process since there are no locks. Almost no function in NodeJS directly performs I/O, so the process never blocks [75]. Because nothing blocks, scalable systems are very reasonable to develop in NodeJS. It also comes with a built-in package manager, which is Node Package Manager (NPM) [8]. NPM allows installing packages easily with a command. Directories of Node applications contain the folder node_ modules where the application is using all packages. Packages are organized in the package.json file by listing all dependencies that are necessary to run the application. The package.json file is important when downloading NodeJS applications from the Internet that do not contain the node_modules folder. The required packages can then be installed by running npm install in this directory. Express[9] is one of the most popular packages installed with NPM. It is a minimal and flexible NodeJS web application framework that provides a robust set of features for web and mobile applications [77]. With Express creating a robust API is quick and easy [77]. An Express application contains at least two files. Firstly, the previously mentioned package.json. Secondly, an Express application also contains a server.js file which is the entry point for the application [76]. Listing 4.8 shows the server.js file for the application.

[8] `npmjs.com/`, last accessed on: 04-09-2021.
[9] `expressjs.com`, last accessed on: 04-09-2021.

```
import express from 'express';
import 'babel-polyfill';
import cors from 'cors';
import env from './env';
import latexRoute from './app/routes/latexRoute';
import analyticsRoute from './app/routes/analyticsRoute';

const app = express();

// Add middleware for parsing URL encoded bodies
app.use(cors());
// Add middleware for parsing JSON and urlencoded data
app.use(express.urlencoded({ extended: false }));
app.use(express.json());

app.use('/api/v1', latexRoute);
app.use('/api/v1', analyticsRoute);


app.listen(env.port).on('listening', () => {
  console.log(`Application is live on ${env.port}`);
});


export default app;
```

Listing 4.8: server.js file for the application

Two different routes have been defined for the application (see lines 16 and 17). The first route is latexRoute.js which includes two end-points. Listing 4.9 shows the end-points.

```
1  import express from 'express';
2
3  import { createPDF } from '../controllers/latexController';
4
5  const router = express.Router();
6
7  // latex Routes
8  router.post('/latex', createPDF);
9  router.get('/latex/download', (req, res) => res.download('./output.pdf'));
10
11
```

Listing 4.9: latexRoute.js file for the application

The first end-point (see line 8) calls the createPDF function, which is implemented in latexController.js. The function is responsible for generating a PDF file based on the client request parameters under the root folder. It uses node-latex[10] package which is a wrapper for generating PDFs with LaTeX[11] in NodeJS. Afterward, the generated PDF file is served with the end-point in line 9. Use case examples are given in Section 4.3.

[10] `https://github.com/saadq/node-latex`, last accessed on: 04-09-2021.
[11] `https://www.latex-project.org/`, last accessed on: 04-09-2021.

The second route is analyticsRoute.js which includes two end-points. Listing 4.10 shows the end-points.

```
1  import express from 'express';
2
3  import { runReport, runDownloadReport }
4  from '../controllers/analyticsController';
5
6  const router = express.Router();
7
8  // latex Routes
9  router.get('/report', runReport);
10 router.get('/report/events', runEventReport);
11
12
13 export default router;
14
15
```

Listing 4.10: analyricsRoute.js file for the application

The first end-point (see line 9) is calling runReport function, which is implemented in the analyticsController.js. The end-point returns a JSON object. An example response can be seen in Listing 4.11.

```
1  [
2    {
3      pageTitle: "Recommender System for Scaling Agile Frameworks",
4      screenPageViews: "1792",
5      userEngagementDuration: "22492",
6    },
7    {
8      pageTitle: "Scaling Agile Hub",
9      screenPageViews: "90",
10     userEngagementDuration: "352",
11   },
12   {
13     pageTitle: "Visualization Pattern  Speed to Market",
14     screenPageViews: "10",
15     userEngagementDuration: "13919",
16   },
```

```
  {
    pageTitle: "Stakeholders  Development Team",
    screenPageViews: "2",
    userEngagementDuration: "1",
  },
  {
    pageTitle:
      "Concerns  Ensuring that the development phases are clearly
      separated and executed in an iterative fashion",
    screenPageViews: "1",
    userEngagementDuration: "70",
  },
  {
    pageTitle: "Evolution of Scaling Agile Frameworks",
    screenPageViews: "1",
    userEngagementDuration: "6",
  },
  {
    pageTitle: "Large Scale Agile Development Patterns",
    screenPageViews: "1",
    userEngagementDuration: "4938",
  },
  {
    pageTitle: "Principles  Strictly separate build and run stages",
    screenPageViews: "1",
    userEngagementDuration: "47",
  },
];


```

Listing 4.11: An example JSON response for the report end-point

The second end-point in Listing 4.10 (see line 10) is calling runEventReport function which is implemented in the analyticsController.js. The end-points provide the events that the frontend application has logged. An example response for the end-point can be seen in Listing 4.12.

```
[
  { eventName: "page_view", eventCount: "1888" },
  { eventName: "user_engagement", eventCount: "1202" },
  { eventName: "session_start", eventCount: "174" },
  { eventName: "first_visit", eventCount: "32" },
  { eventName: "screen_view", eventCount: "10" },
  { eventName: "full_catalog", eventCount: "5" },
];


```

Listing 4.12: An example JSON response for the report/events end-point

The analyticsController.js is using google-analytics[12] package to retrieve the data which is reported by the frontend application. Listing 4.13 shows the implementation of the runEventReport function.

```
import { errorMessage, successMessage, status } from "../helpers/status";

const propertyId = process.env.PROPERTY_ID;
const { AlphaAnalyticsDataClient } = require("@google-analytics/data");

const runEventReport = async (req, res) => {
  // Creates a client
const client = new AlphaAnalyticsDataClient();
  const [response] = await client.runReport({
    entity: {
      propertyId: propertyId,
    },
    dateRanges: [
      {
        startDate: "2020-03-31",
        endDate: "today",
      },
    ],
```

[12] https://github.com/googleapis/nodejs-analytics-data, last accessed on: 04-09-2021.

```
    dimensions: [
      {
        name: "eventName",
      },
    ],
    metrics: [
      {
        name: "eventCount",
      },
    ],
  });
  let data = [];

  response.rows.forEach((row) => {
    let child = {};
    data.push(
      new Promise((resolve, reject) => {
        child["eventName"] = row.dimensionValues[0].value;
        child["eventCount"] = row.metricValues[0].value;
        resolve(child);
      })
    );
  });

  return res.status(status.success).send(await Promise.all(data));
};

export { runEventReport };

```

Listing 4.13: runEventReport function

Moreover, Figure 4.4 shows the overall folder structure for the NodeJS application.

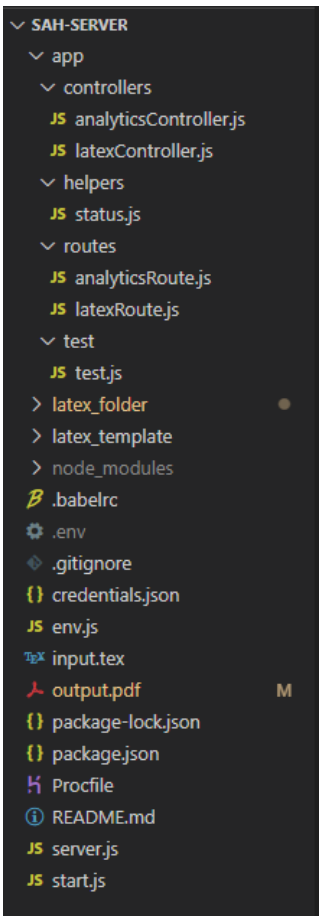


Figure 4.4: Overall structure for the NodeJS application

4.3 System architecture

The following section provides a brief insight into the technical architecture of the application. The system architecture is kept simple. It consists of the React single-page application serving as the frontend for the user, which is communicating with the SocioCortex and NodeJS REST-APIs via HTTP calls. Figure 4.5 illustrates a high-level overview of the architecture.

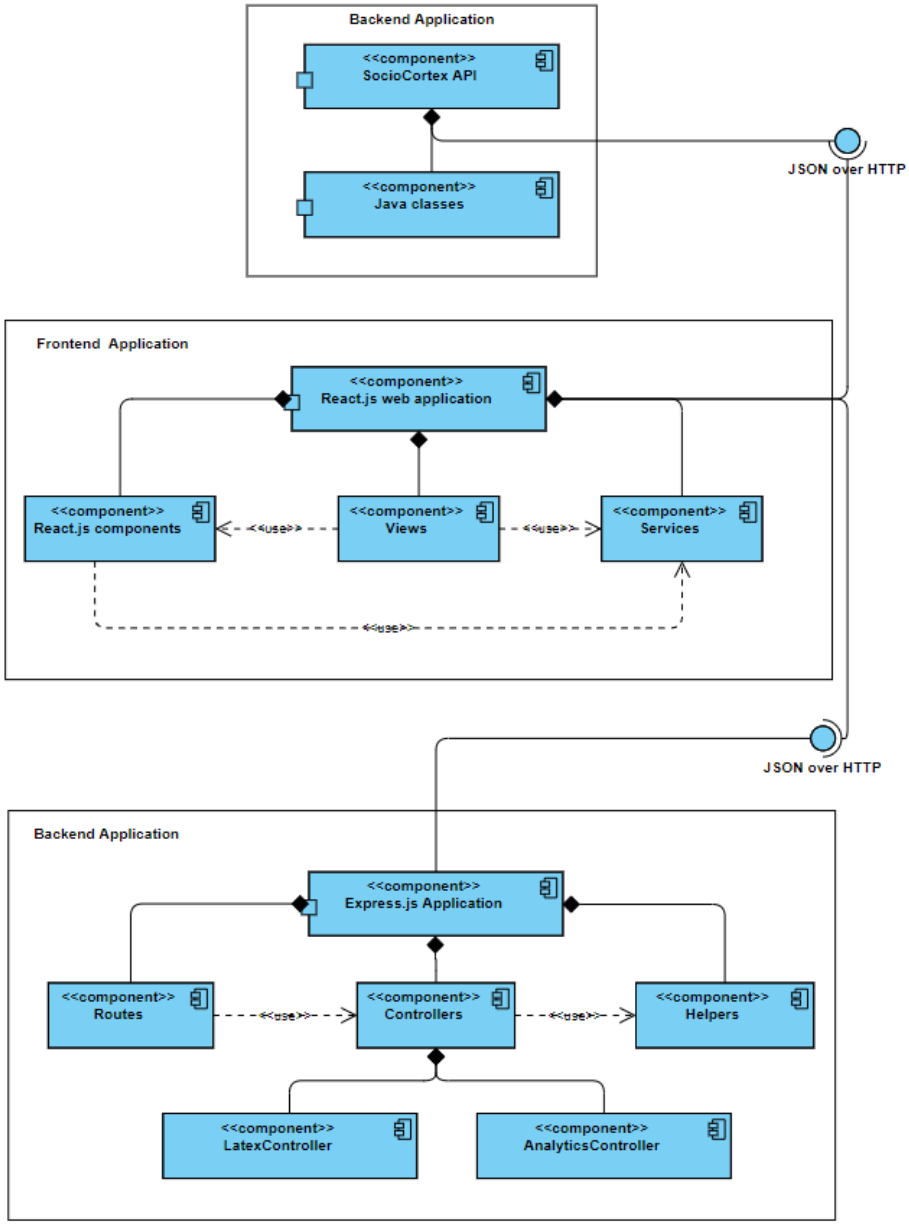


Figure 4.5: High-level overview of the system architecture

4.4 Main views and core features

This section includes the core features of the application based on the main views in which they are used within the application. The features were carefully designed and implemented based on the challenges identified in Chapter 4 and continuously discussed, adjusted, and extended over the course of the prototype implementation. Social design principles are also taken into account that Kraut and Resnick introduced in chapter "Encouraging Contribution to Online Communities" [22] when designing the application. The landing page of the application can be seen in Figure 4.6. The application includes the following core features as well as other supportive features:

- User access management
- Badge system
- Activity feed
- Pattern visualization
- PDF export

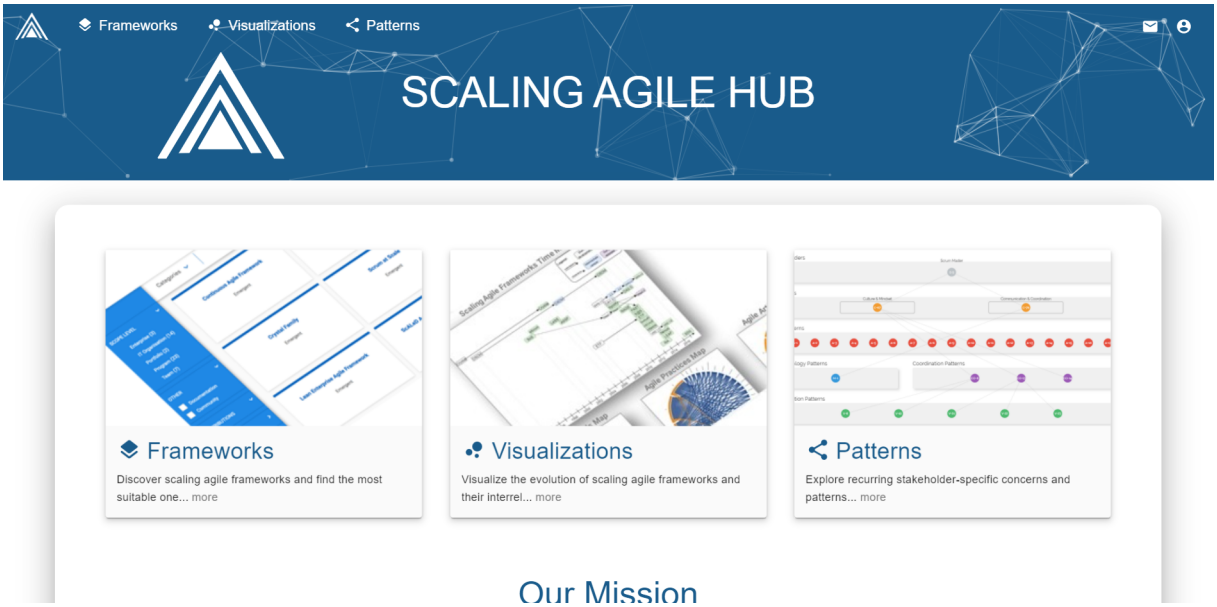


Figure 4.6: Landing page of the application

4.4.1 User Access Management

User access management feature is implemented to address the following challenges that are identified by Henninger et al. [21]:

- Software pattern validation
- Tracking software pattern variants and duplicates
- Updating software pattern knowledge

To be able to address the challenges, the following functionalities were implemented.

Register and Login

The user registration screen can be seen in Figure 4.7. After the registration, the system automatically enables the user by using a magic link that SocioCortex generates.

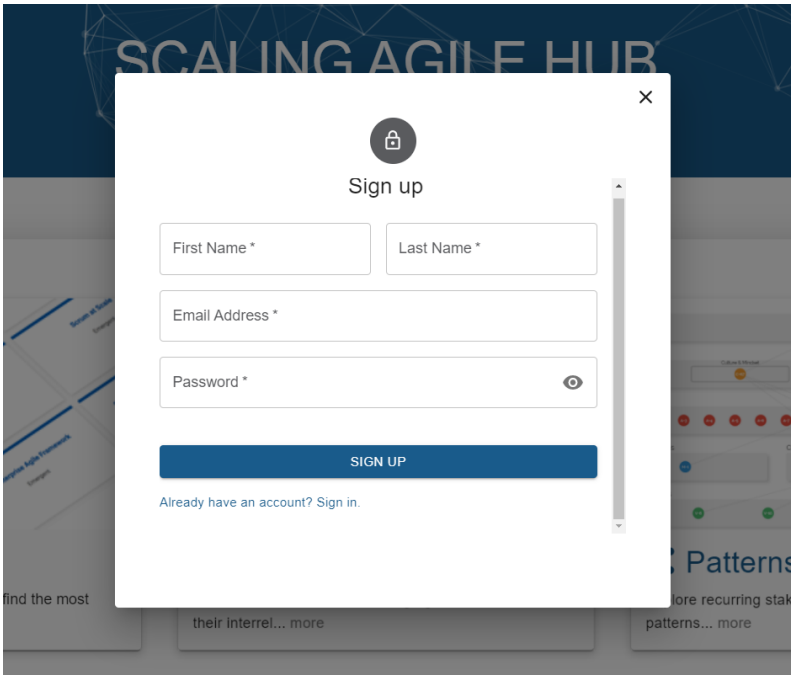


Figure 4.7: Sign Up Page of the application

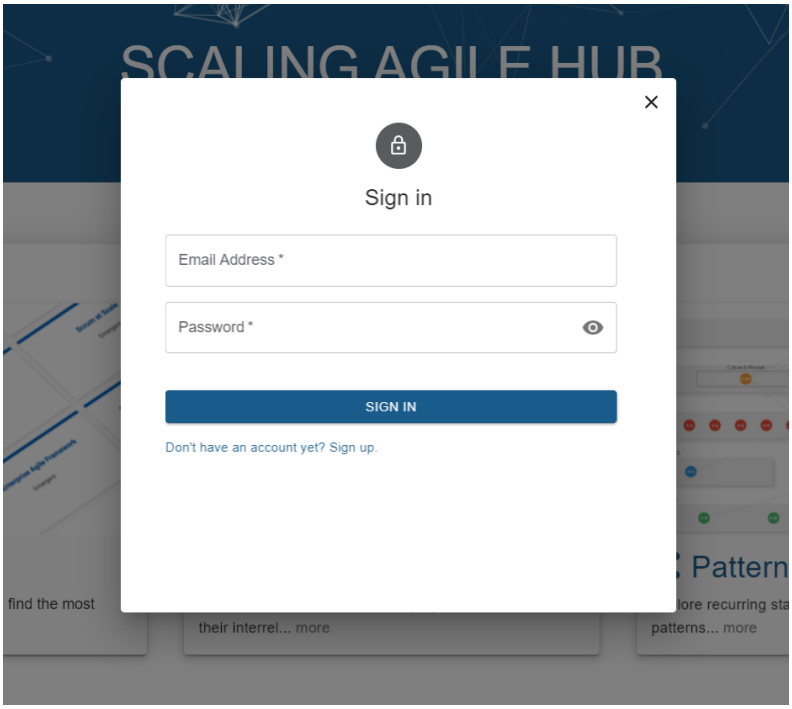


Figure 4.8: Sign In Page of the application

Create and Edit Pattern

The pattern creation feature allows admins to easily add new patterns, as shown in Figure 4.9.

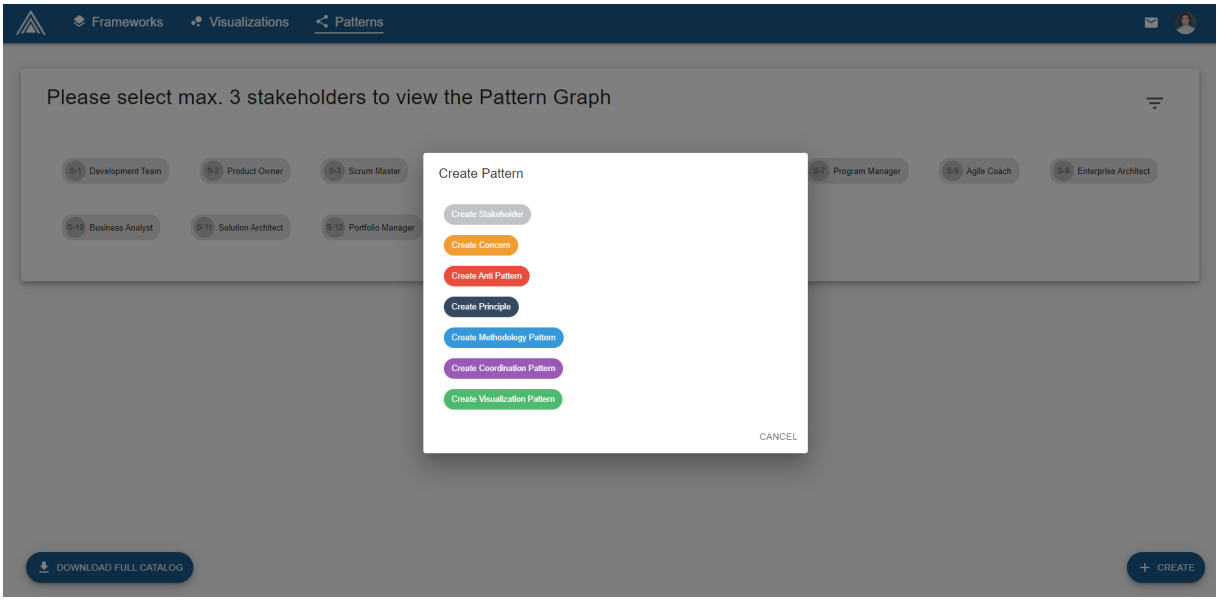


Figure 4.9: Dialog for creating a new pattern

After selecting the type of pattern, the user is prompted to provide information regarding the pattern. Figure 4.10 shows a dialog for creating a new visualization pattern.

The edit pattern feature addresses the "updating software pattern knowledge" challenge mentioned by Henninger et al. [21]. Figure 4.11 shows the edit pattern view. When there is a change regarding the patterns, it can easily be updated by admins.

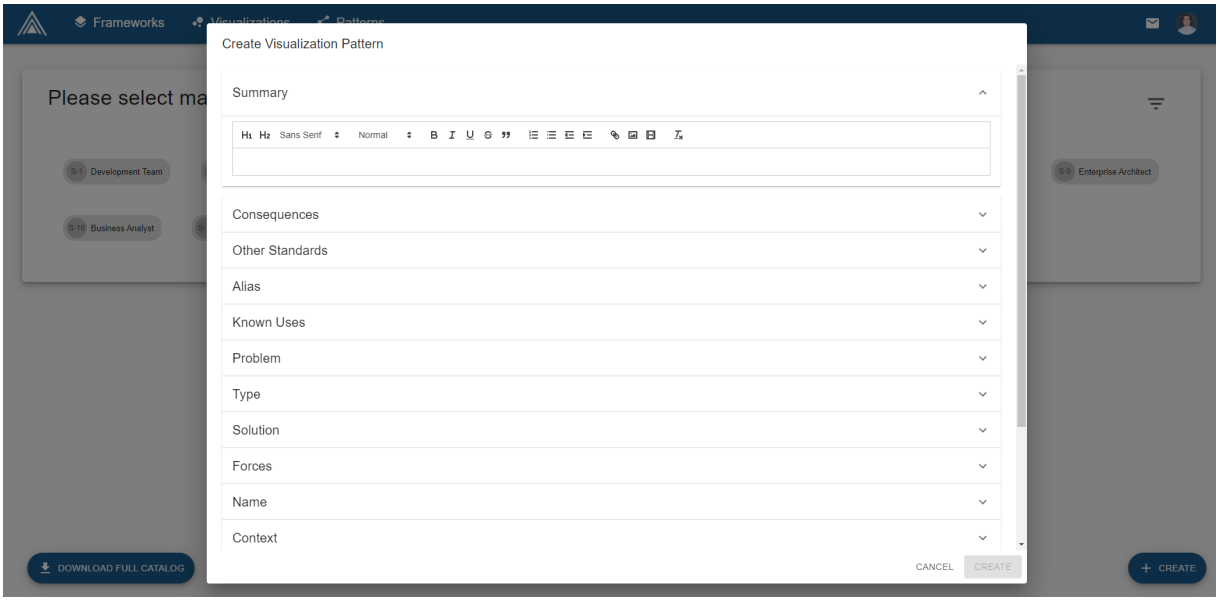


Figure 4.10: Dialog for creating a new visualization pattern

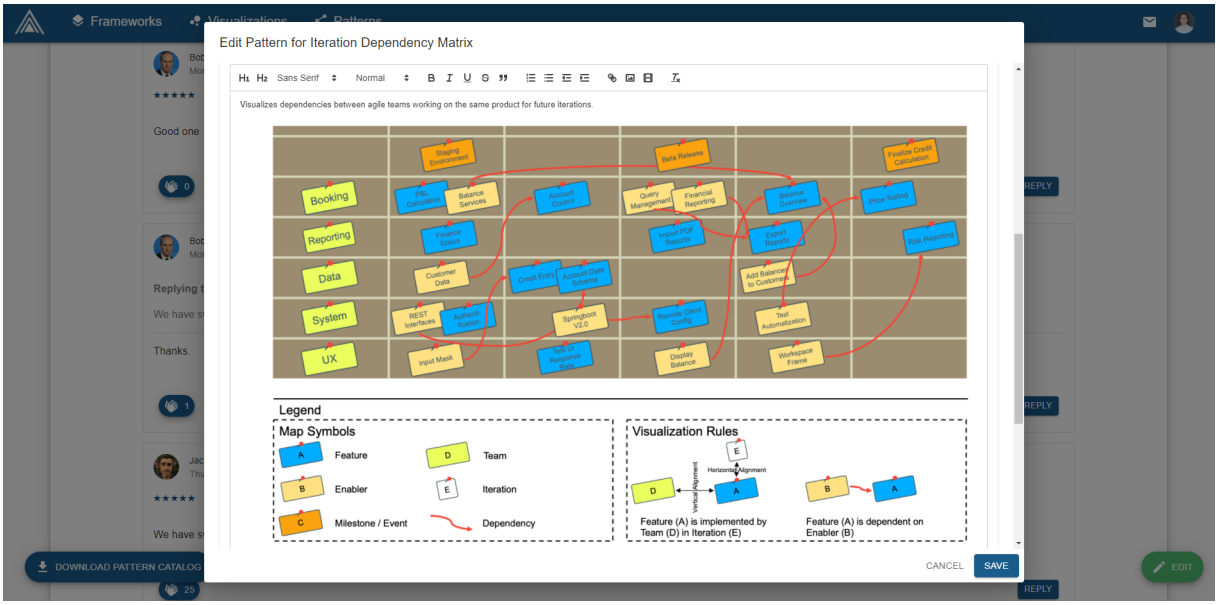


Figure 4.11: Edit pattern view

Add and Up-vote Feedback

The adding feedback feature enables pattern assessment. Users can state their opinions freely using this feature which can be seen in Figure 4.12. It is also possible to up-vote feedback, which makes feedback differentiated from each other. Figure 4.13 shows the feedback view for a pattern. Users can reply to the feedback given by other users, which can be seen in Figure 4.14. This also enables a feedback assessment.

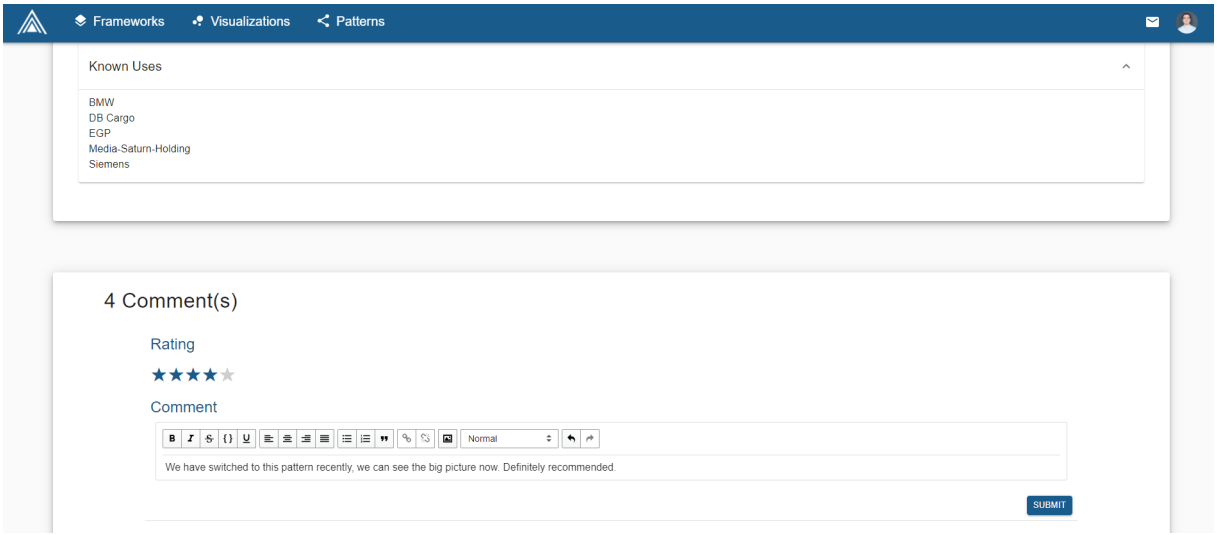


Figure 4.12: Add feedback view

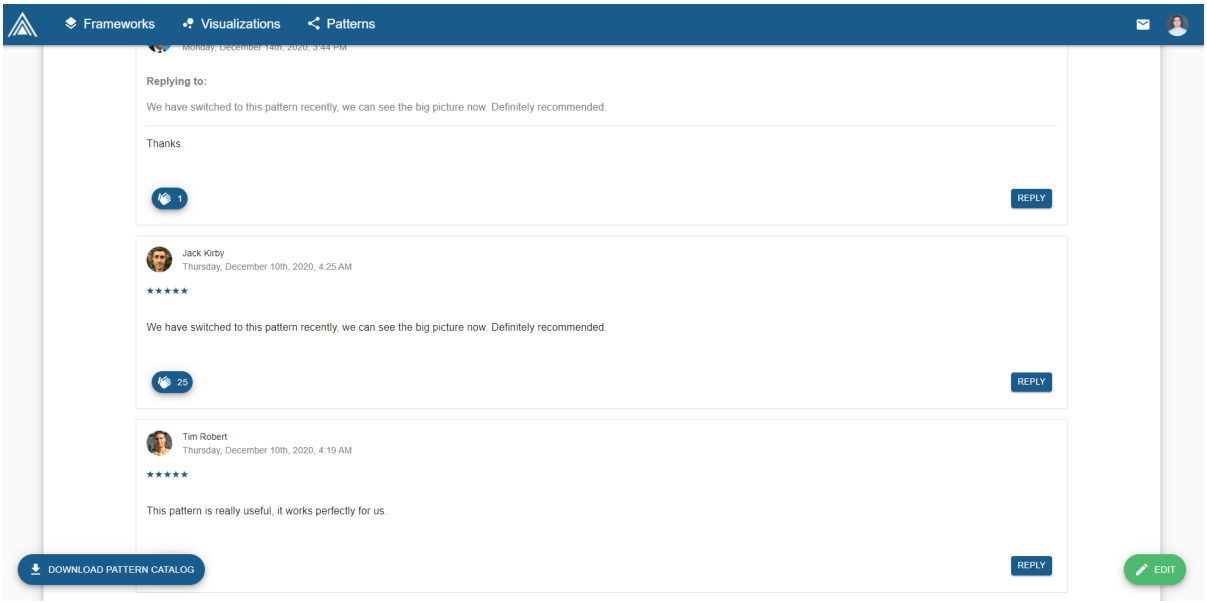


Figure 4.13: Pattern feedback view

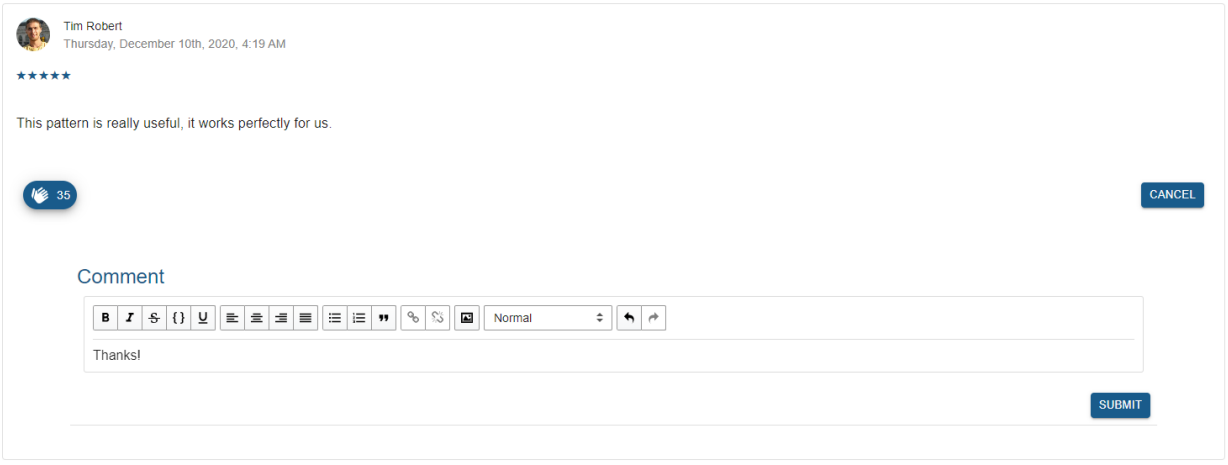


Figure 4.14: Reply feedback view

4.4.2 Badge System

Since the value of the application is dependent on the active usage and contribution of its users, we incorporate insights from social science into the design of our features. The badge system feature is inspired by the following design claim by Kraut and Resnick [21]:

> *Rewards, whether in the form of status, privileges or material benefits - motivate contributions. (Design claim #23 in [21])*

Achieving a new, higher badge can already be seen as a sort of status. Three types of badges are defined for the application as bronze, silver, and gold. Badges are given based on the user's reputation points, and the points are earned based on the user activities on the platform. For example, getting an up-vote for the feedback which the user gives. Figure 4.15 shows a profile page that has a golden badge.

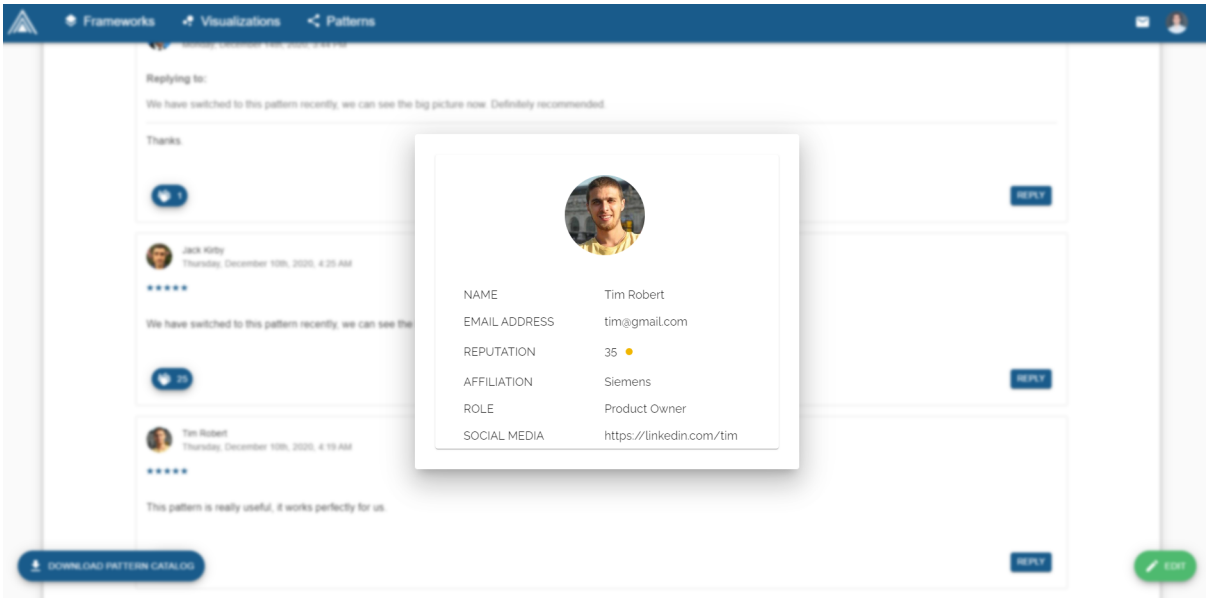


Figure 4.15: Profile page view

4.4.3 Activity Feed

The latest activities from the users are shown on the landing page. The activity feed feature is inspired by the following design claim by Kraut and Resnick [21] which can be seen in Figure 4.16:

Motivate contributions by showing social similarities (Design claim #11 in [21])

This design claim fits well with the landing page's activity feed since social similarities such as affiliation and role are displayed.

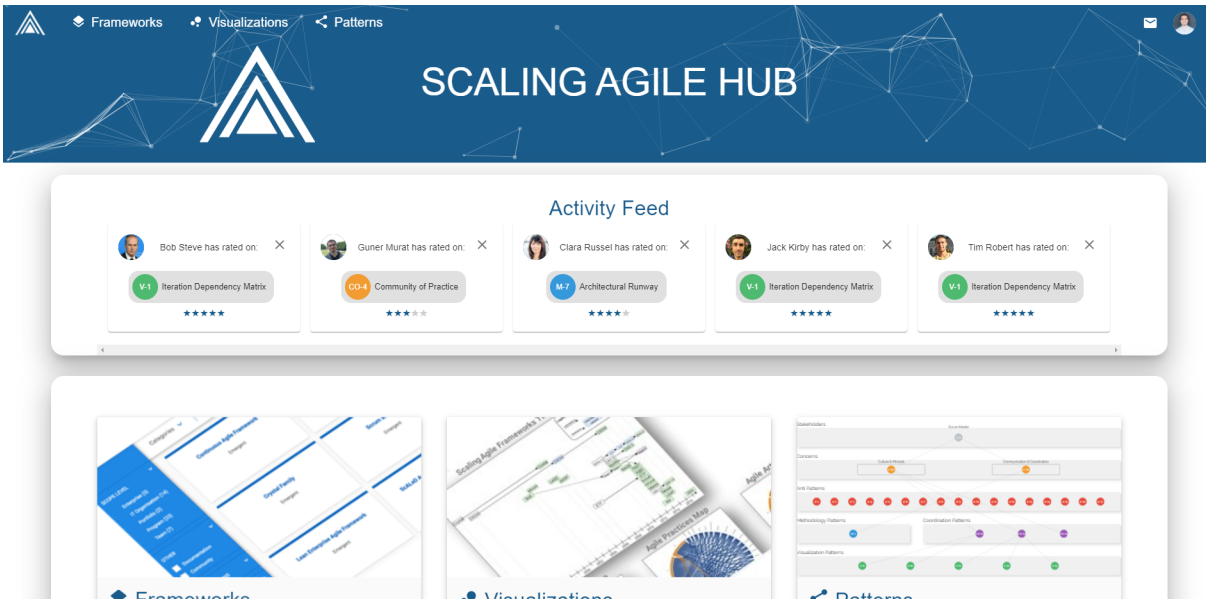


Figure 4.16: Activity feed view

4.4.4 Pattern visualization

Lack of standard pattern form is another challenge identified by Henninger et al. [21]. Pattern visualization features are implemented to be able to address this challenge. On the patterns page, shown in Figure 4.17, users can select stakeholders to see the related pattern graph.

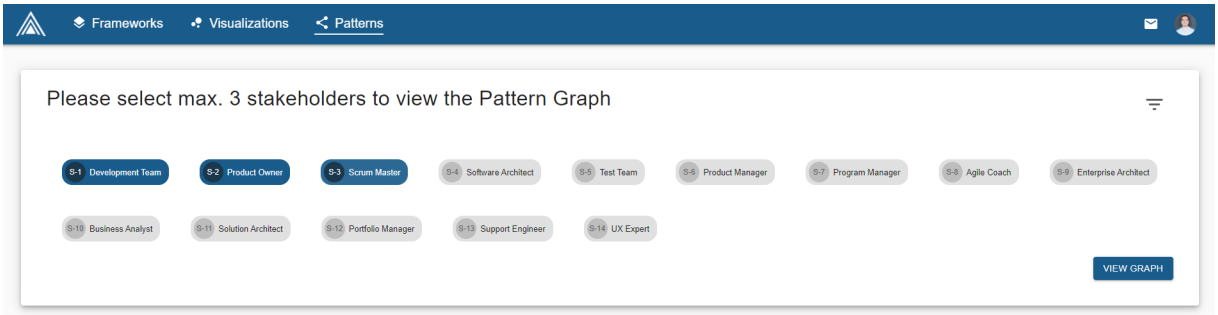


Figure 4.17: Stakeholder selection view

After the selection, the Large-Scale Agile Development Pattern Graph is presented, as shown in Figure 4.18.

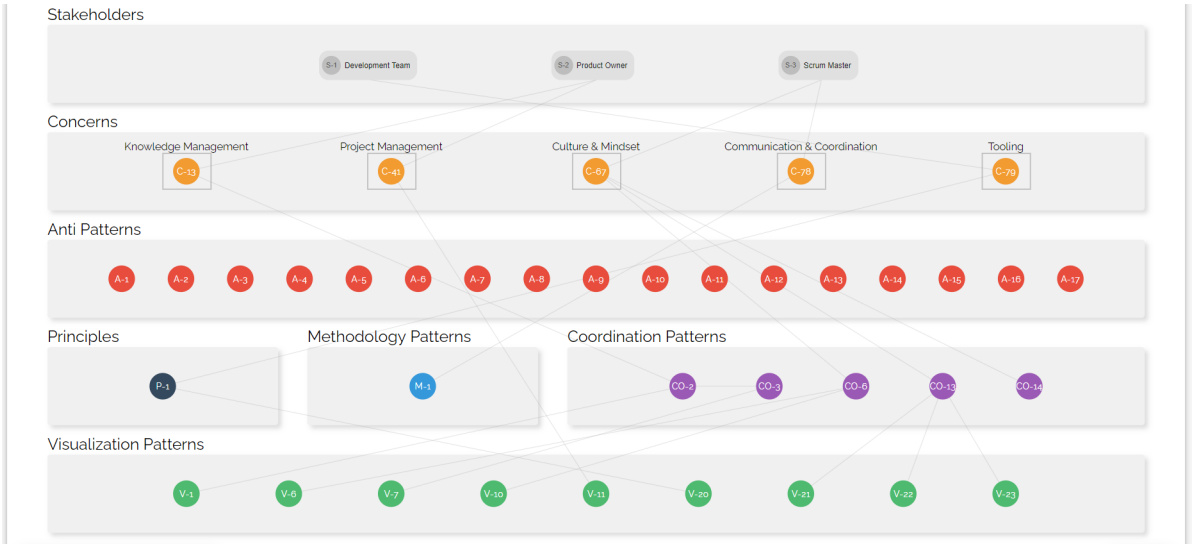


Figure 4.18: Large-Scale Agile Development Pattern Graph

It is also possible to see a related pattern graph based on the highlighted stakeholder, shown in Figure 4.19.

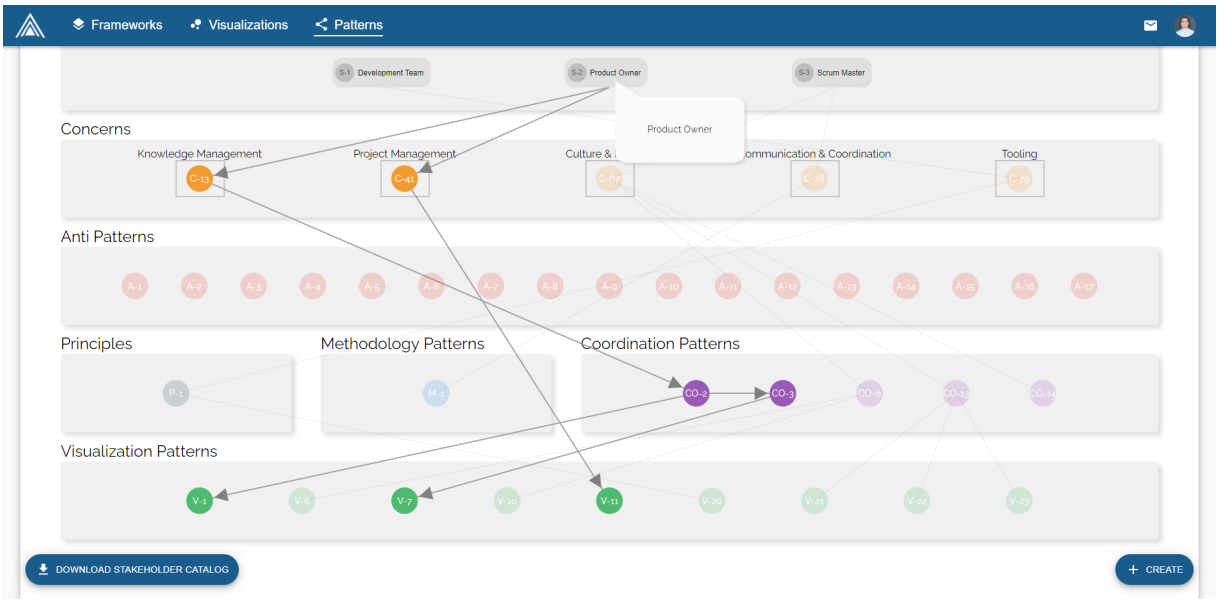


Figure 4.19: Highlighted pattern graph

With this feature, the selected stakeholder's general overview can be visualized, and even further, each node can be visualized specifically to get more insight. E.g. Figure 4.20 represents the view when "V-1 (Visualization pattern - Iteration Dependency Matrix)" is selected from the pattern graph.

On this page, below the pattern graph, all the necessary information can be found regarding the selected pattern. Figure 4.21 shows the Iteration Dependency Matrix's information view representing a visualization pattern.

Once the tabs are expanded by clicking on the arrow button, which is on the right side of the labels, detailed information can be seen. Figure 4.22 shows an example view for expanded tabs.

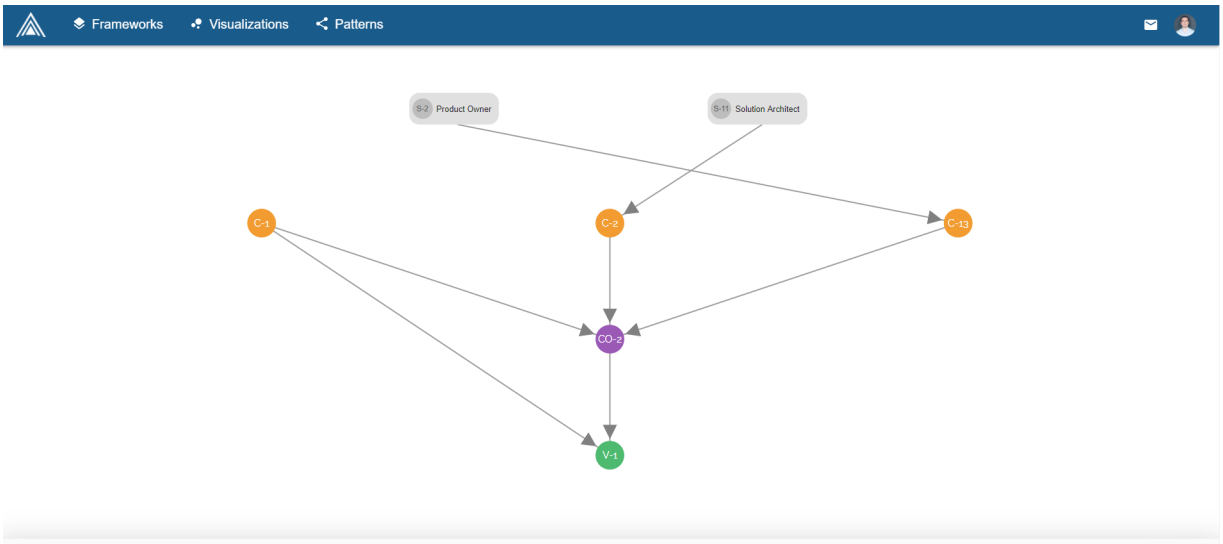


Figure 4.20: V-1 pattern view (Visualization pattern - Iteration Dependency Matrix)

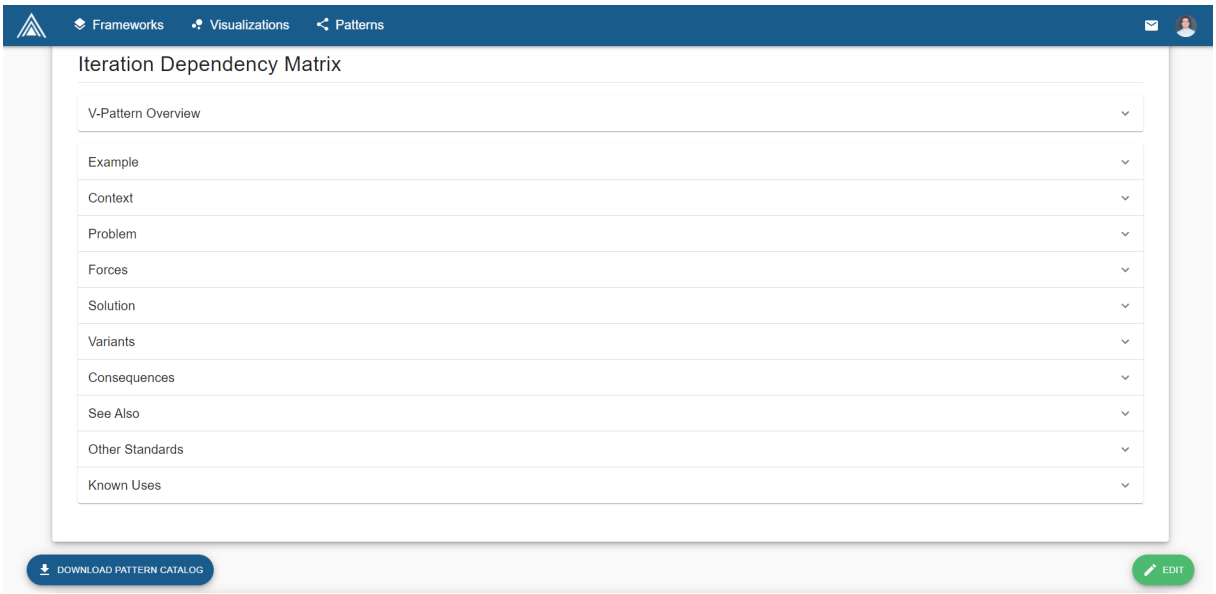


Figure 4.21: V-1 Information view (Visualization pattern - Iteration Dependency Matrix)

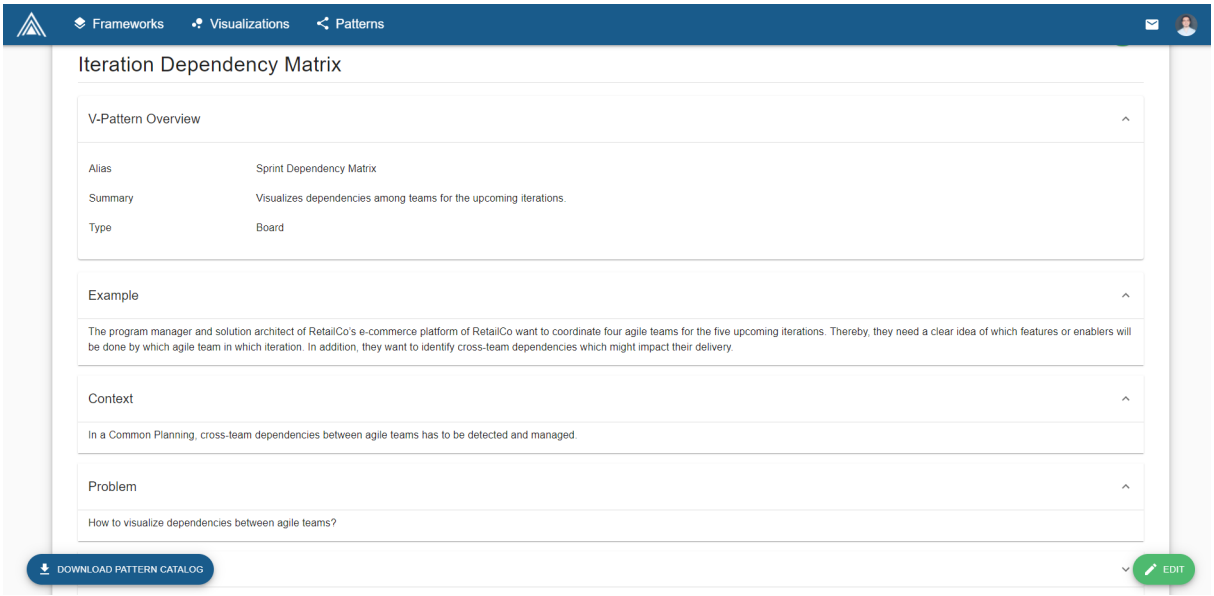


Figure 4.22: Expanded information view (Visualization pattern - Iteration Dependency Matrix)

4.4.5 PDF Export

The PDF export functionality is implemented to address the "Electronic Accessibility" challenge identified by Henninger et al. [21]. It is aimed to make the information available even offline. Thus, three types of download options are provided:

- Full catalog download which can be seen in Figure 4.23
- Stakeholder catalog download which can be seen in Figure 4.19
- Pattern catalog download which can be seen in Figure 4.22

The full catalog download option enables users to download all the pattern-related information included in the platform. It is also possible to download the data based on the selected stakeholders and patterns, which means that other information will be excluded.

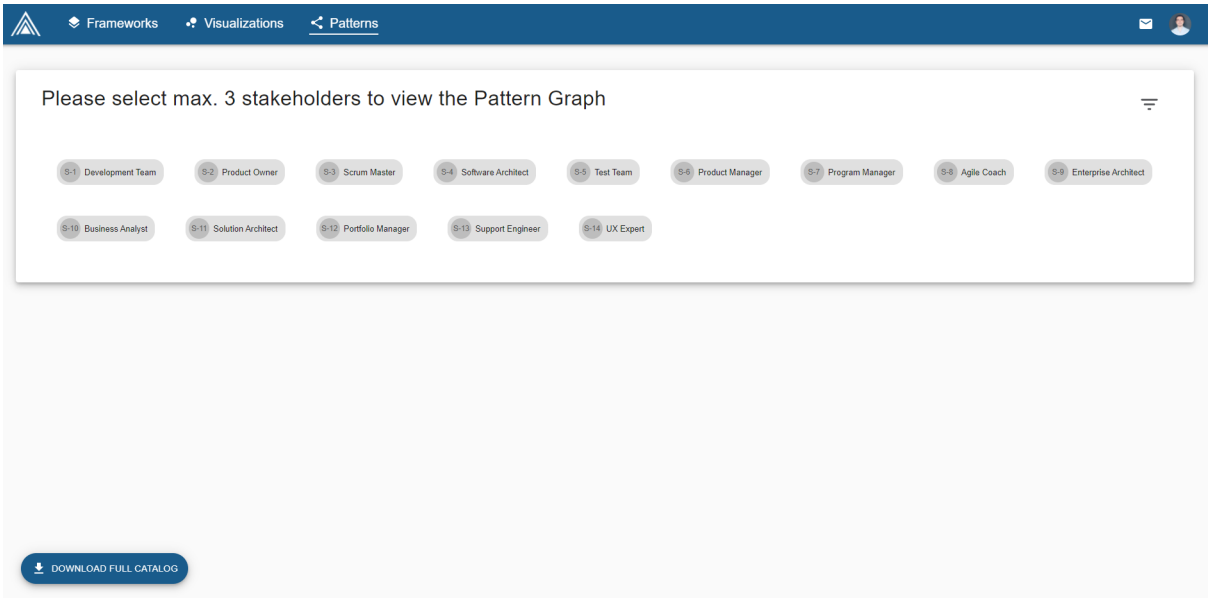


Figure 4.23: Full catalog download view

5 Evaluation

In this chapter, we share the analytics results of the prototype. Detailed evaluation is difficult since the platform lacks patterns data. Thus, it is prioritized to increase the amount of pattern data on the platform. It is possible to see how potential users are behaving on the platform. There are four types of data visualizations in terms of the analytics data:

- User count and engagement time view
- Demographic view
- Page view
- Event view

5.1 User Count and Engagement Time View

The user count and engagement time view shows the total number of users that are interacting with the platform, new users count, and average engagement time within a given time. The related data for the platform can be seen in Figure 5.1.

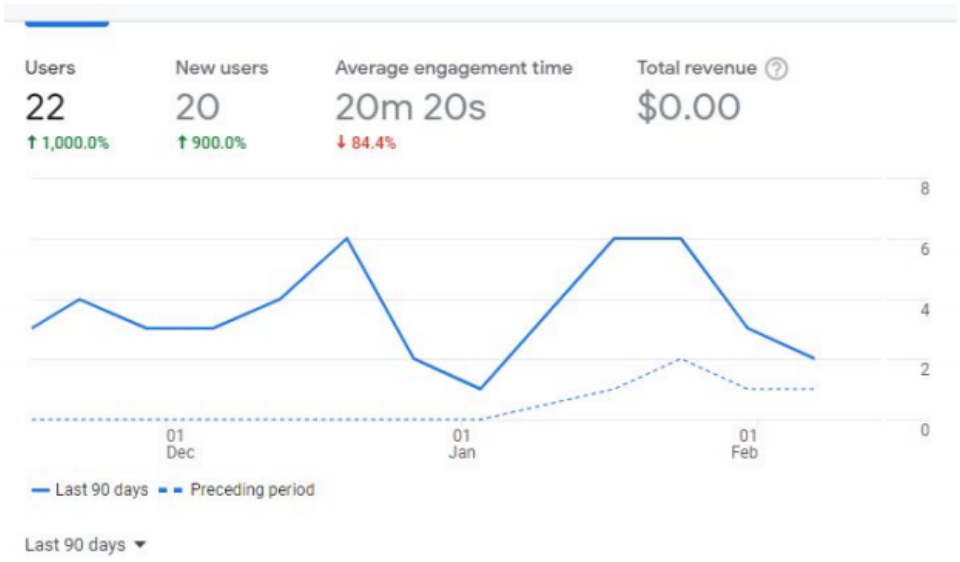


Figure 5.1: User count and engagement time view

5.2 Demographic View

The demographic view presents the location of the users that they connect to the platform. Figure 5.2 shows the demographic data for the platform.

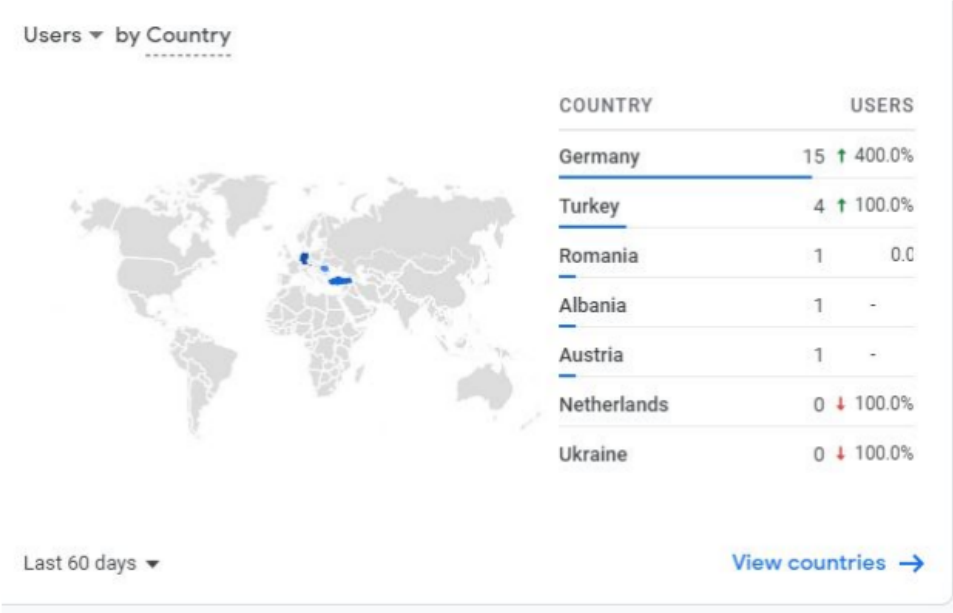


Figure 5.2: User count and engagement time view

Further, it is possible to see detailed information on the demographic data such as new users and the users' average engagement time based on the countries. Detailed demographic information can be seen in Figure 5.3.

Figure 5.3: Detailed demographic view

5.3 Page View

The page view presents views based on the page title. It is possible to get further details such as unique user scrolls on the pages, event counts, and average engagement time. Figure 5.4 shows the page view for the platform.

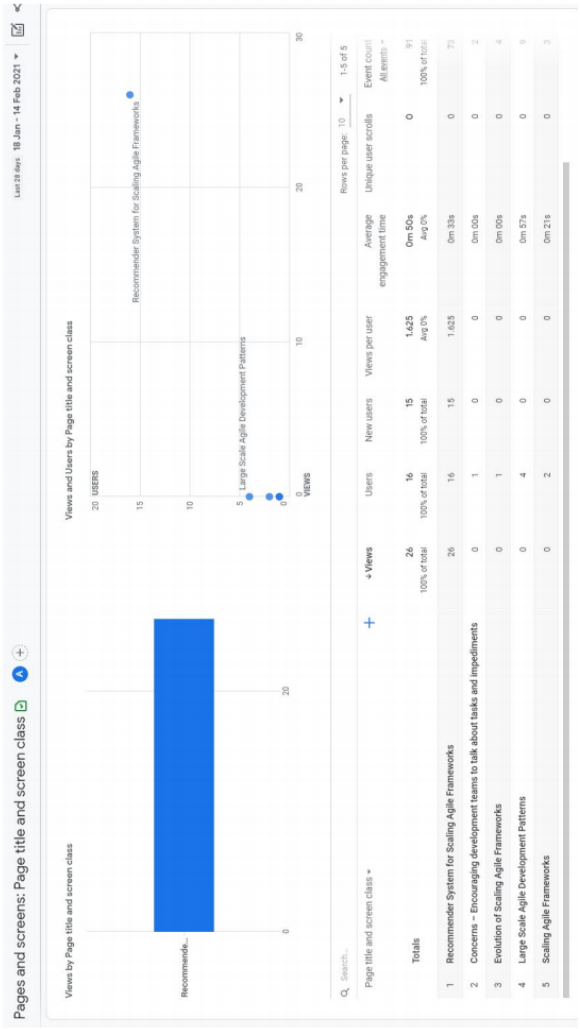


Figure 5.4: Page view

5.4 Event View

The event view shows event-based data such as session start information, first visit information, and many more events that have been reported. The event view can be seen in Figure 5.5 for the application.

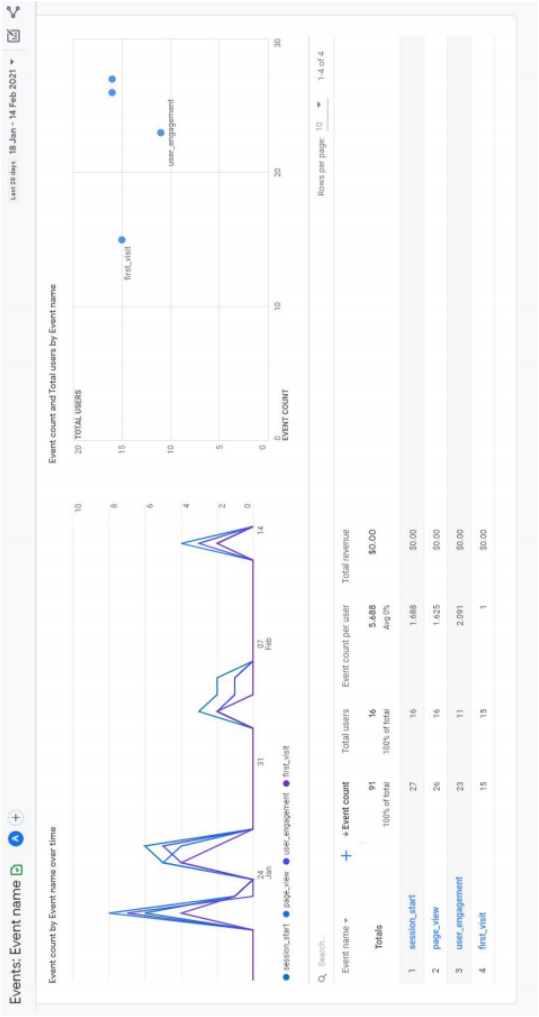


Figure 5.5: Event view

6 Discussion

This chapter summarizes key findings and provides a critical reflection that discusses this master's thesis's potential limitations.

6.1 Key Findings

In the following, we describe and summarize the key findings of this master's thesis.

A web application addresses typical challenges of traditional pattern formats and communities
Throughout the empirical work by Henninger et al. [21], a set of challenges identified for federating pattern collections into a more integrated and interconnected body of knowledge. The challenges are heavily biased toward combining patterns in a distributed electronic format utilizing Web technologies. The identified challenges must be met to achieve this goal, such as electronic accessibility of the patterns, lack of standard pattern forms, software pattern validation, and updating software pattern knowledge.

Existing pattern communities
- **do not provide patterns regarding large-scale agile development**
- **exhibit shortcomings related to the creation of online communities**
- **show limitations related to the effective identification and validation of patterns**

We identified throughout our research different pattern communities, such as Scrum-PLoP [78], EuroPloP [79], Hillside Group [80] and the ILDE (Integrated Learning Design Environment) [81] which includes both a template for editing patterns and pattern-based learning design tools. While they all provide pattern knowledge, there are some problems, such as lack of guidance, finding large-scale development patterns difficult, lack of design concepts to promote contribution, and lack of interactivity.

Promotion of desirable user behavior is possible through certain design concepts in online communities
Kraut and Resnick documented design claims based on Social Science and Psychology researches in the chapter "Encouraging Contribution to Online Communities" [22]. It is also stated that the promotion of desirable behaviors is possible through certain design concepts in communities. These design concepts are also applicable to an online community. Based on these design claims, the badge system and activity feed features were implemented. One good example is Stack Overflow that makes extensive use of badges. The use of badge incentives is a new trend in online social websites, and it has a significant effect on user engagement and participation [82].

A web application can be valuable for supporting pattern communities related to large-scale agile development
Patterns are available in several publishing mediums, from books to proceedings to Web sites. Although 31% of the software patterns are locked in book format (proceedings, journal, book), 69% are electronically accessible on the Web. However, less than half (44%) of the Web-accessible patterns are represented using structured text such as HTML (10% of patterns) or XML. The rest is available through PS/PDF/Word files [21]. Thus, a web application with features presented in the thesis can provide valuable support for supporting pattern communities at large-scale development. So far, the proposed web application mainly aims to provide a social and interactive tool and increase the accessibility of patterns. The prototypical web application also provides the basis for future enhancements, such as providing deviated patterns that organizations use.

6.2 Limitations

This section discusses limitations and threats to the validity of the thesis. First of all, a limitation is the limited time frame of the thesis available for conducting the research, which does not allow for a longer evaluation and actual usage of the solution artifacts over multiple months. Another key limitation is the static mapping of the pattern catalog. Since the pattern catalog is manually created and needs to be updated when there is a change, some changes regarding the pattern knowledge might not be adopted immediately to the downloaded pattern catalogs. Another limitation is the lack of user feedback. We aimed to counteract this limitation by implementing analytics module into our solution artifacts' design along with existing literature and related work.

7 Conclusion and Future Work

This chapter summarizes the thesis based on the research questions and presents an outlook for further research.

7.1 Summary

The following section summarizes the answers to the research questions presented in Section 1.2.

> ***Research question 1:*** *What are the challenges of establishing pattern communities?*

Patterns are generally disseminated in dispersed collections through a range of publication mediums with little to no technical assistance. Potential pattern users are having trouble learning what patterns are available and where, and how to use them. The amount of patterns and range of pattern forms continues to increase [21]. Throughout the existing literature review, six challenges were identified as electronic accessibility of patterns, lack of standard pattern forms, inter-pattern relationships, pattern validation, tracking pattern variants, duplicates, and updating pattern knowledge. Additionally, we have spotted design concepts presented in Building Successful Online Communities: Evidence-Based Social Design book [22] by Kraut and Resnick promote desirable behavior in online communities.

> ***Research question 2:*** *How can a prototypical web implementation support the establishment of pattern communities?*

A prototypical web implementation was implemented to address the identified challenges by the research question 1. The derived challenges and design concepts are mapped as features for the web implementation, such as user access management, pattern visualization, PDF export, badge system, and activity feed.

> ***Research question 3:*** *How can the prototypical web implementation be improved in the future?*

We have compared the prototypical web implementation with other online communities such as ScrumPLoP [78], EuroPloP [79], Hillside Group [80] and the ILDE

(Integrated Learning Design Environment) [81]. Based on the literature and comparison results, we have identified possible future extensions such as enabling users to upload patterns on the platform by adding a review and approve the process, implementing pattern deviations feature to know how companies are using the patterns, enhancing badge system by giving rewards based on the points that are earned on the platform and integrating other design concepts that might motive users.

7.2 Future Work

Due to this master's thesis's limited time frame, surveys to evaluate the protoype could not be conducted with industry partners. The valuable feedback from the conducted expert interviews and surveys can be one of the crucial next steps for further work. Another aspect of future work is the generalization of the results. Firstly, applying and evaluating the prototypical web implementation after publishing it to the community could provide helpful insights into further refining and adjusting the approach and the demonstrated web application. Moreover, the article with the title of Pattern-Based Design Research – An Iterative Research Method Balancing Rigor and Relevance that is published by Matthes et al. [83] could be analyzed to determine how pattern variants can be integrated into the platform that companies use.

Bibliography

[1] W. J. Orlikowski. "Improvising Organizational Transformation Over Time: A Situated Change Perspective." In: *Information Systems Research* 7.1 (1996), pp. 63–92. DOI: 10.1287/isre.7.1.63.

[2] C. Fuchs and T. Hess. "Becoming agile in the digital transformation: The process of a large-scale agile transformation." In: *39th International Conference On Information Systems, San Francisco*. 2018.

[3] P. Weill and S. Woerner. "Thriving in an increasingly digital ecosystem." In: *MIT Sloan Management Review* 56.4 (2015), p. 27.

[4] W. K. B. Sherehiy and J. Layer. "A review of enterprise agility: Concepts, frameworks, and attributes." In: *International Journal of industrial ergonomics* 37.5 (2007), pp. 445–460.

[5] Ö. Uludağ, M. Kleehaus, X. Xu, and F. Matthes. "Investigating the Role of Architects in Scaling Agile Frameworks." In: *21st IEEE International Enterprise Distributed Object Computing Conference, EDOC 2017, Quebec City, QC, Canada, October 10-13, 2017*. 2017, pp. 123–132. DOI: 10.1109/EDOC.2017.25.

[6] *Factors that impact implementing an agile software development methodology*. 2007, pp. 82–86. DOI: 10.1109/SECON.2007.342860.

[7] A. B. E. Overby and V. Sambamurthy. "Enterprise agility and the enabling role of information technology." In: *European Journal of Information Systems* 15.2 (2006), pp. 120–131.

[8] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al. *Manifesto for Agile Software Development*. http://agilemanifesto.org. 2001.

[9] C. W. K. Petersen. "The effect of moving from a plan-driven to an incremental software development approach with agile practices." In: *Empirical Software Engineering* 15.6 (2010), pp. 654–693.

[10] K. Beck. "Extreme Programming Explained: Embrace Change." In: (1999).

[11] P. Kettunen. "Extending Software Project Agility with New Product Development Enterprise Agility." In: *Softw. Process. Improv. Pract.* 12.6 (2007), pp. 541–548.

[12] K. Schwaber and M. Beedle. "Agile Software Development with Scrum." In: (2002).

[13] P. Kettunen. "Extending Software Project Agility with New Product Development Enterprise Agility." In: *Softw. Process* 12.6 (Nov. 2007), pp. 541–548. ISSN: 1077-4866. DOI: 10.1002/spip.v12:6.

[14] T. Dingsøyr, N. B. Moe, R. Tonelli, S. Counsell, Ç. Gencel, and K. Petersen, eds. *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation - XP 2014 International Workshops, Rome, Italy, May 26-30, 2014, Revised Selected Papers*. Vol. 199. Lecture Notes in Business Information Processing. Springer, 2014. ISBN: 978-3-319-14357-6.

[15] T. Dingsøyr and N. B. Moe. "Research challenges in large-scale agile software development." In: *ACM SIGSOFT Software Engineering Notes* 38.5 (2013), pp. 38–39. DOI: 10.1145/2507288.2507322.

[16] N. B. Moe and T. Dingsøyr. "Emerging research themes and updated research agenda for large-scale agile development: a summary of the 5th international workshop at XP2017." In: *Proceedings of the XP2017 Scientific Workshops, Cologne, Germany, May 22 - 26, 2017*. ACM. 2017, 14:1–14:4. DOI: 10.1145/3120459.3120474.

[17] Ö. Uludag, P. Philipp, A. Putta, M. Paasivaara, C. Lassenius, and F. Matthes. "Revealing the State-of-the-Art in Large-Scale Agile Development: A Systematic Mapping Study." In: *CoRR* abs/2007.05578 (2020).

[18] M. Alqudah and R. Razali. "A Review of Scaling Agile Methods in Large Software Development." In: *International Journal on Advanced Science, Engineering and Information Technology* 6.6 (2016), pp. 828–837. ISSN: 2088-5334. DOI: 10.18517/ijaseit.6.6.1374.

[19] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. *Manifesto for Agile Software Development*. Retrieved April 13, 2016, from http://agilemanifesto.org/. 2001.

[20] Ö. Uludağ, N.-M. Harders, and F. Matthes. "Documenting Recurring Concerns and Patterns in Large-Scale Agile Development." In: *24th European Conference on Pattern Languages of Programs*. 2019.

[21] S. Henninger and V. Corrêa. "Software Pattern Communities: Current Practices and Challenges." In: *Proceedings of the 14th Conference on Pattern Languages of Programs*. PLOP '07. Monticello, Illinois, USA: Association for Computing Machinery, 2007. ISBN: 9781605584119. DOI: 10.1145/1772070.1772087.

[22] R. E. Kraut and P. Resnick. *Building successful online communities: Evidence-based social design*. Mit Press, 2012.

[23] A. R. Hevner, S. T. March, J. Park, and S. Ram. "Design Science in Information Systems Research." In: *MIS Quarterly* 28.1 (2004), pp. 75–106.

[24] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. "A Design Science Research Methodology for Information Systems Research." In: *Journal of Management Information Systems* 24.3 (2007), pp. 45–77. DOI: 10.2753/MIS0742-1222240302.

[25] P. Hohl, J. Klünder, A. van Bennekum, R. Lockard, J. Gifford, J. Münch, M. Stupperich, and K. Schneider. "Back to the future: origins and directions of the "Agile Manifesto" – views of the originators." In: *Journal of Software Engineering Research and Development* (2018).

[26] A. Cockburn. *Agile Software Development: The Cooperative Game*. Agile Software Development Series. Pearson Education, 2006. ISBN: 9780321630070.

[27] P. Runeson and M. Höst. "Guidelines for conducting and reporting case study research in software engineering." In: *Empirical software engineering* 14.2 (2009), p. 131.

[28] N. Abbas, A. Gravell, and G. Wills. "Historical Roots of Agile Methods: Where Did "Agile Thinking" Come From?" In: vol. 9. June 2008. ISBN: 978-3-540-68254-7. DOI: 10.1007/978-3-540-68255-4_10.

[29] A. Stellman and J. Greene. *Learning Agile: Understanding Scrum, XP, Lean, and Kanban*. O'Reilly, 2014. ISBN: 9781449331924.

[30] P. Serrador and J. Pinto. "Does Agile work? — A quantitative analysis of agile project success." In: *International Journal of Project Management* 33 (Mar. 2015), pp. 1040–1051. DOI: 10.1016/j.ijproman.2015.01.006.

[31] M. Coram and S. Bohner. "The impact of Agile Methods on software project management." In: May 2005, pp. 363–370. ISBN: 0-7695-2308-0. DOI: 10.1109/ECBS.2005.68.

[32] A. B. M. Moniruzzaman and S. Hossain. "Comparative Study on Agile software development methodologies." In: (July 2013).

[33] H. Takeuchi and I. Nonaka. "The new new product development game." In: *Harvard business review* 64.1 (1986), pp. 137–146.

[34] D. Maximini. *The Scrum Culture: Introducing Agile Methods in Organizations*. Springer Publishing Company, Incorporated, 2015. ISBN: 3319118269.

[35] K. Schwaber. "Scrum development process." In: *Business object design and implementation*. Springer, 1997, pp. 117–134.

[36] H. F. Cervone. "Understanding agile project management methods using Scrum." In: *OCLC Systems Services* 27 (Feb. 2011), pp. 18–22. DOI: 10.1108/10650751111106528.

[37] *Scrumorg Scrum Framework*. https://scrumorg-website-prod.s3.amazonaws.com/drupal/2021-01/Scrumorg-Scrum-Framework-tabloid.pdf. (Accessed on 04/11/2021).

[38] K. Schwaber. *Agile Project Management with Scrum*. Developer Best Practices. Pearson Education, 2004. ISBN: 9780735637900.

[39] C. Keith. *Agile Game Development with Scrum*. Addison-Wesley Signature Series (Cohn). Pearson Education, 2010. ISBN: 9780321670281.

[40] K. Rubin. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley signature series. Addison-Wesley, 2012. ISBN: 9780137043293.

[41] R. Sindhgatta, N. C. Narendra, and B. Sengupta. "Software evolution in agile development: a case study." In: *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*. ACM. 2010, pp. 105–114. DOI: 10.1145/1869542.1869560.

[42] W. Hayes, S. Miller, M. A. Lapham, E. Wrubel, and T. Chick. *Agile metrics: Progress monitoring of agile contractors*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2014.

[43] D. Anderson. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010. ISBN: 9780984521401.

[44] M. O. Ahmad, J. Markkula, and M. Oivo. "Kanban in software development: A systematic literature review." In: *2013 39th Euromicro conference on software engineering and advanced applications*. IEEE. 2013, pp. 9–16. DOI: 10.1109/SEAA.2013.28.

[45] M. Ikonen, P. Kettunen, N. Oza, and P. Abrahamsson. "Exploring the sources of waste in kanban software development projects." In: *36th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE. 2010, pp. 376–381. DOI: 10.1109/SEAA.2010.40.

[46] V. Mahnic. "Improving software development through combination of scrum and kanban." In: *Recent Advances in Computer Engineering, Communications and Information Technology, Espanha* (2014), pp. 281–288.

[47] J. Sutherland and K. Schwaber. *The definitive guide to scrum: The rules of the game.* http://www.scrum.org/scrum-guides. 2013 (Accessed on 10/16/2019).

[48] M. Yilmaz and R. V. O'Connor. "A Scrumban integrated gamification approach to guide software process improvement: a Turkish case study." In: *Tehnički vjesnik* 23.1 (2016), pp. 237–245. DOI: 10.17559/TV-20140922220409.

[49] A. Reddy. *The Scrumban [R]Evolution: Getting the Most Out of Agile, Scrum, and Lean Kanban*. Agile Software Development Series. Pearson Education, 2015. ISBN: 9780134077628.

[50] M. Stoica, B. Ghilic-Micu, M. Mircea, and C. Uscatu. "Analyzing Agile Development from Waterfall Style to Scrumban." In: *Informatica Economica* 20.4 (2016). DOI: 10.12948/issn14531305/20.4.2016.01.

[51] E. Corona and F. E. Pani. "A review of lean-Kanban approaches in the software development." In: *WSEAS transactions on information science and applications* 10.1 (2013), pp. 1–13.

[52] M. Poppendieck and M. A. Cusumano. "Lean software development: A tutorial." In: *IEEE software* 29.5 (2012), pp. 26–32. DOI: 10.1109/MS.2012.107.

[53] T. Dingsøyr, T. Fægri, and J. Itkonen. "What Is Large in Large-Scale?A Taxonomy of Scale for Agile Software Development." English. In: *Product-Focused Software Process Improvement Lecture Notes in Computer Science*. Ed. by A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch, and M. Raatikainen. 8892nd ed. Springer International Publishing, 2014, pp. 273–276. ISBN: 978-3-319-13834-3.

[54] T. Dingsøyr, N. B. Moe, T. E. Fægri, and E. A. Seim. "Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation." In: *Empirical Software Engineering* 23.1 (2018), pp. 490–520. DOI: 10.1007/s10664-017-9524-2.

[55] L. Lagerberg, T. Skude, P. Emanuelsson, K. Sandahl, and D. Ståhl. "The Impact of Agile Principles and Practices on Large-Scale Software Development Projects: A Multiple-Case Study of Two Projects at Ericsson." In: *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE. 2013, pp. 348–356. DOI: 10.1109/ESEM.2013.53.

[56] J. Vlietland and H. van Vliet. "Towards a governance framework for chains of Scrum teams." English. In: *Information and Software Technology* 57.1 (2015), pp. 52–65. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2014.08.008.

[57] K. Petersen and C. Wohlin. "The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study." In: *Empirical Software Engineering* 15 (Dec. 2010), pp. 654–693. DOI: 10.1007/s10664-010-9136-6.

[58] N. B. Moe, D. Smite, A. Sablis, A.-L. Börjesson, and P. Andréasson. "Networking in a large-scale distributed agile project." In: *2014 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14, Torino, Italy, September 18-19, 2014*. ACM. 2014, 12:1–12:8. DOI: 10.1145/2652524.2652584.

[59] M. A. Babar. "An exploratory study of architectural practices and challenges in using agile software development approaches." In: *2009 Joint Working IEEE/IFIP Conference on Software Architecture European Conference on Software Architecture*. 2009, pp. 81–90. DOI: 10.1109/WICSA.2009.5290794.

[60] D. Leffingwell. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. 1st. Addison-Wesley Professional, 2011. ISBN: 0321635841.

[61] K. Dikert, M. Paasivaara, and C. Lassenius. "Challenges and success factors for large-scale agile transformations: A systematic literature review." In: *Journal of Systems and Software* 119 (2016), pp. 87–108. DOI: 10.1016/j.jss.2016.06.013.

[62] D. Badampudi, S. A. Fricker, and A. M. Moreno. "Perspectives on Productivity and Delays in Large-Scale Agile Projects." In: *International Conference on Agile Software Development*. Springer. 2013, pp. 180–194. DOI: 10.1007/978-3-642-38314-4_13.

[63] P. Kettunen and M. Laanti. "Combining agile software projects and large-scale organizational agility." In: *Software Process: Improvement and Practice* 13.2 (2008), pp. 183–193. DOI: 10.1002/spip.354.

[64] M. Alqudah and R. Razali. "A review of scaling agile methods in large software development." In: *International Journal on Advanced Science, Engineering and Information Technology* 6.6 (2016), pp. 828–837. DOI: 10.18517/ijaseit.6.6.1374.

[65] Ö. Uludağ, M. Kleehaus, C. Caprano, and F. Matthes. "Identifying and structuring challenges in large-scale agile development based on a structured literature review." In: *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE. 2018, pp. 191–197. DOI: 10.1109/EDOC.2018.00032.

[66] Ö. Uludağ and F. Matthes. "Identifying and Documenting Recurring Concerns and Best Practices of Agile Coaches and Scrum Masters in Large-Scale Agile Development." In: 2019.

[67] Ö. Uludağ and F. Matthes. "Large-Scale Agile Development Patterns for Enterprise and Solution Architects." In: *Proceedings of the European Conference on Pattern Languages of Programs 2020*. EuroPLoP '20. Virtual Event, Germany: Association for Computing Machinery, 2020. ISBN: 9781450377690. DOI: 10.1145/3424771.3424895.

[68] *Published Patterns*. https://sites.google.com/a/scrumplop.org/published-patterns/home. Accessed: 2020-12-30.

[69] *sebis TU München : SocioCortex - Model-Based Collaboration Environment*. https://wwwmatthes.in.tum.de/pages/13uzffgwlh8z4/SocioCortex. (Accessed on 02/19/2021).

[70] F. Matthes, C. Neubert, and A. Steinhoff. "Hybrid Wikis: Empowering Users to Collaboratively Structure Information." In: *ICSOFT (1)*. Ed. by M. J. E. Cuaresma, B. Shishkov, and J. Cordeiro. SciTePress, 2011, pp. 250–259. ISBN: 978-989-8425-76-8.

[71] T. Reschenhofer, M. Bhat, A. Hernandez-Mendez, and F. Matthes. "Lessons Learned in Aligning Data and Model Evolution in Collaborative Information Systems." In: *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. 2016, pp. 132–141.

[72] *Basic Types in MxL*. https://sociocortex.com/tutorial/2015/12/01/mxl02/. (Accessed on 02/20/2021).

[73] *React – A JavaScript library for building user interfaces*. https://reactjs.org/. (Accessed on 02/25/2021).

[74] K. Chinnathambi. *Learning React*. Boston: Addison-Wesley, 2017. ISBN: 978-0-13-454631-5.

[75] *About | Node.js*. https://nodejs.org/en/about/. (Accessed on 02/26/2021).

[76] M. Satheesh. *Web development with MongoDB and NodeJS : build an interactive and full-featured web application from scratch using Node.js and MongoDB*. Birmingham, UK: Packt Publishing, 2015. ISBN: 978-1-78528-752-7.

[77] *Express - Node.js web application framework*. https://expressjs.com/. (Accessed on 02/27/2021).

[78] *Scrum Pattern Community*. http://www.scrumplop.org/. (Accessed on 03/14/2021).

[79] *EuroPLoP*. https://europlop.net/. (Accessed on 03/14/2021).

[80] *The Hillside Group - A group dedicated to design patterns. Home of the patterns library.* https://hillside.net/. (Accessed on 03/14/2021).

[81] *Welcome to ILDEplus! - ILDE*. https://ilde.upf.edu/pg/lds/firststeps/. (Accessed on 03/14/2021).

[82] R. Gharibi and M. Malekzadeh. "Gamified Incentives: A Badge Recommendation Model to Improve User Engagement in Social Networking Websites." In: *International Journal of Advanced Computer Science and Applications* 8 (Jan. 2017). DOI: 10.14569/IJACSA.2017.080533.

[83] S. Buckl, F. Matthes, A. W. Schneider, and C. M. Schweda. "Pattern-Based Design Research – An Iterative Research Method Balancing Rigor and Relevance." In: *Design Science at the Intersection of Physical and Virtual Design*. Ed. by J. vom Brocke, R. Hekkala, S. Ram, and M. Rossi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 73–87. ISBN: 978-3-642-38827-9.