

INSTITUT FÜR INFORMATIK
der Ludwig-Maximilians-Universität München

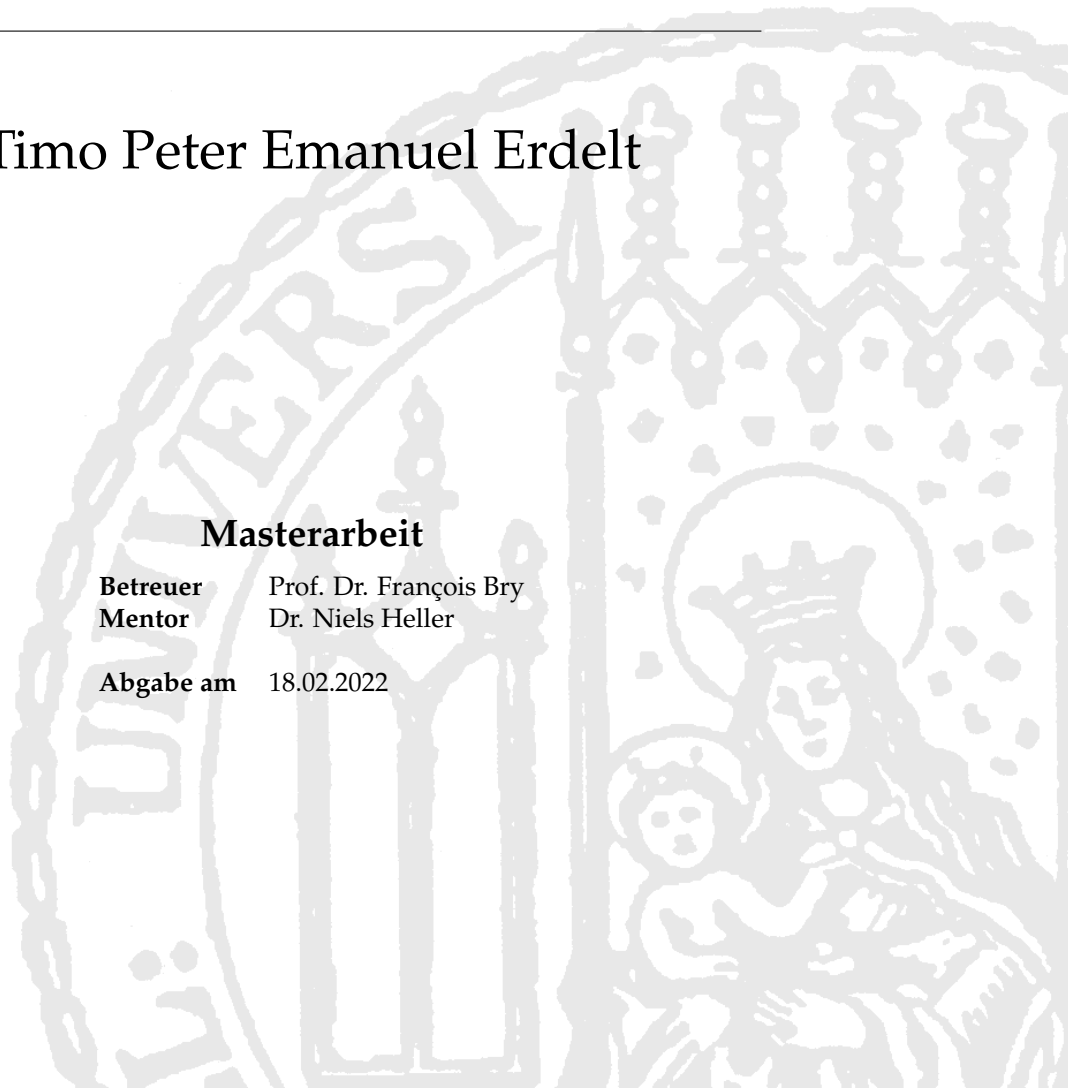
INTELLIGENT TASK MANAGEMENT AND VISUALISATION

Timo Peter Emanuel Erdelt

Masterarbeit

Betreuer Prof. Dr. François Bry
Mentor Dr. Niels Heller

Abgabe am 18.02.2022



Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst habe und keine anderen als die angegebenen Hilfsmittel verwendet habe.

München, den 18.02.2022

Timo Peter Emanuel Erdelt

Abstract

Nowadays, Kanban board-based software solutions are used by millions of teams and individuals for managing and visualising projects and their corresponding tasks.

In commonly used Kanban board implementations, a board consists of multiple vertical lists of tasks arranged in one horizontal list. As deeper nesting is not intended, the visualisation's expressiveness is limited for more complex projects. Also, current Kanban board-based applications lack quantified information regarding the project's progress which support project scheduling.

This work presents the prototype of a board-based task management software that allows arranging tasks in arbitrarily nested horizontal or vertical lists to address these issues. Furthermore, a symbolic intelligence that creates proposals for scheduling tasks under given temporal constraints is implemented to support users in planning their projects.

The novel features of the prototype were evaluated in a user study with 13 participants. In the study, participants preferred and created project visualisations with a significantly greater nesting depth than the current standard. The evaluation indicates that removing the bound of nesting depth can improve the visualisation of projects in board-based tools. Furthermore, the participants' manual scheduling would have resulted in a significantly later project finish date with more due dates missed than the scheduling proposed by the symbolic intelligence. Lastly, the evaluation has also shown the symbolic intelligence to be welcomed and perceived positively. These findings indicate, that the symbolic intelligence can support users in scheduling their projects. Overall the evaluation shows the potential of the new approaches presented in this work to improve current Kanban board-based task management implementations.

Zusammenfassung

Kanban-Board-basierte Softwarelösungen helfen heutzutage Millionen von Teams und Einzelpersonen bei der Verwaltung und Visualisierung von Projekten und den dazugehörigen Aufgaben.

In gängigen Implementierungen besteht ein Board aus einer horizontalen Liste, welche vertikale Listen mit Aufgaben nebeneinander anordnet. Da eine tiefere Verschachtelung nicht vorgesehen ist, ist die Aussagekraft der Visualisierung für komplexe Projekte begrenzt. Außerdem fehlt es bestehenden Implementierungen an quantitativen Informationen über den Fortschritt eines Projekts, welche für die Projektplanung relevant sind.

Mit dem Ziel die Visualisierung zu verbessern, erlaubt es der für diese Arbeit entwickelte Prototyp einer Board-basierten Aufgabenverwaltungs-Software, Aufgaben in beliebig verschachtelten horizontalen oder vertikalen Listen anzuordnen. Um Nutzerinnen und Nutzer bei der zeitlichen Planung von Projekten zu unterstützen, wurde zudem eine symbolische Intelligenz implementiert, welche Vorschläge für die Planung von Aufgaben unter zuvor konfigurierten zeitlichen Einschränkungen erstellt und hilft, den Stand des Projekts besser einzuschätzen.

Die neuartigen Konzepte des Prototypen wurden mithilfe einer Studie mit 13 Teilnehmenden evaluiert. In der Studie bevorzugten und erstellten die Teilnehmenden Projektvisualisierungen mit einer signifikant größeren Verschachtelungstiefe als der derzeitige Standard. Die Ergebnisse deuten darauf hin, dass die Aufhebung der Beschränkung der Verschachtelungstiefe die Visualisierung von Projekten in Board-basierten Softwarelösungen verbessern kann. Des Weiteren führte die manuelle Aufgabenplanung der Teilnehmenden im Vergleich zur von der symbolischen Intelligenz vorgeschlagenen Planung zu einem signifikant späteren Projektabschlusszeitpunkt sowie mehr verpassten Fälligkeitsdaten. Die Implementierung der symbolischen Intelligenz wurde zudem begrüßt und positiv wahrgenommen. Diese Ergebnisse deuten daher darauf hin, dass die symbolische Intelligenz die zeitliche Planung von Projekten unterstützen kann.

Insgesamt zeigt die Auswertung, dass die in dieser Arbeit vorgeschlagenen Ansätze das Potential haben, derzeitige Kanban-Board-basierten Aufgabenmanagement-Implementierungen zu verbessern.

Acknowledgments

I would like to thank Professor Dr. François Bry for offering me the opportunity to write my thesis at his chair. Also, I would like to thank him for his much appreciated support throughout the long process of developing the prototype and writing this thesis. Our regular meetings and discussions motivated me and helped me generate and reflect on the ideas presented in this work.

Furthermore, I would like to thank Dr. Niels Heller for providing the thesis project's initial idea and motivation and for his great support and feedback throughout the process, which enabled me to realise the project and write the thesis.

Lastly, I would like to thank all participants of my user study and those who supported me through feedback and advice in developing the prototype and improving my writing.

Contents

1	Introduction	1
2	Related Work	5
2.1	From Lean to Agile	5
2.1.1	Lean approach	5
2.1.2	Agile software development	6
2.2	Kanban	6
2.2.1	Kanban method	6
2.2.2	Kanban board	6
2.2.3	Kanban vs. Scrum	7
2.2.4	Benefits and adoption in software development	7
2.2.5	Kanban in other fields	8
2.2.6	Challenges and areas of improvement	9
2.3	Virtual Kanban board implementations	10
2.3.1	Prototype implementations	10
2.3.2	Consumer products	10
2.3.3	Improvement of existing tools	12
2.4	Data-driven estimation of agile software projects	12
2.4.1	Effort estimation	12
2.4.2	Lead time estimation	13
3	Concept	15
3.1	Goals	15
3.2	Ideation and development process	16
3.3	Features of the prototype	16
3.3.1	User and workspace	16
3.3.2	Elements	17
3.3.3	Resources	20
3.3.4	Events	20
3.3.5	Recommendations: Symbolic intelligence	21
3.4	Design and user experience	21
4	Implementation	23
4.1	Architectural Overview	23
4.2	Server	24
4.2.1	Technology stack	24
4.2.2	Data model	24

4.2.3	Rest API	24
4.2.4	Symbolic intelligence	24
4.2.4.1	Constraint solver	24
4.2.4.2	Data preparation, assumptions and calculations	28
4.3	Web application: Technology stack	29
5	Evaluation	31
5.1	Methodology	31
5.1.1	Study design	31
5.1.2	Procedure	31
5.1.2.1	Intro	31
5.1.2.2	Part 1 - Arbitrarily nested lists	32
5.1.2.3	Part 2 - Symbolic intelligence	33
5.1.2.4	Outro	33
5.2	Participants	36
5.3	Limitations	36
5.4	Measurements	36
5.4.1	Quantitative measures	36
5.4.2	Qualitative measures	37
5.5	Data analysis and results	38
5.5.1	Quantitative data	38
5.5.1.1	Part 1 - Arbitrarily nested lists	38
5.5.1.2	Part 2 - Symbolic Intelligence	42
5.5.1.3	General usability	47
5.5.2	Qualitative data	48
5.5.2.1	Questions	48
5.5.2.2	Observation	49
6	Discussion and future work	51
6.1	Discussion of the user study results	51
6.1.1	Unbounded nesting	51
6.1.2	Symbolic intelligence	52
6.1.3	General usability	53
6.1.4	Outlook	54
6.2	Areas of improvement for the prototype	55
6.2.1	Unbounded nesting	55
6.2.1.1	Kanban method context	55
6.2.1.2	Visualisation of unbounded nesting	55
6.2.1.3	Moving elements in nested situations	56
6.2.2	Scheduling feature	56
6.2.2.1	Visualisation of the scheduling result	57
6.2.2.2	Constraint solver enhancements	57
6.2.2.3	Support through data-driven techniques	58
6.2.2.4	Scheduling everywhere	58
6.2.3	General usability	59
6.2.4	Additional features	59
6.3	Use cases	60
6.3.1	Agile software development	60
6.3.2	Studying at a university	60
6.3.3	Laboratory work organisation	61
7	Conclusion	63

<i>CONTENTS</i>	xi
Appendix	65
Bibliography	71

CHAPTER 1

Introduction

In the complex, fast-moving, and digitised world of the 21st-century, people have to shift their attention between different areas of their life constantly. Being able to self-organise projects in areas like work, studying or one's personal life plays a key part in improving productivity and relieving stress at the same time. While this is already important for individuals, big companies competitiveness heavily depends on effective workflow management. In the 1940s, the so-called Lean approach got introduced in the Japanese car manufacturing industry to optimise business processes and eliminate waste [39]. One of the tools to execute the Lean approach in practice was the Kanban system, a just-in-time scheduling system invented at Toyota [28]. Nowadays, Lean thinking is applied in many global industries and is also considered the origin for the evolution of modern agile software development methodologies such as the Kanban method [9]. Based on the original Kanban system, the Kanban method aims to "visualise the workflow", "limit the work in progress", "measure and manage the flow", and "improve collaboratively" [3]. The Kanban board, which is the tool to execute these principles in practice, can be implemented physically using post-its on a wall or virtually using collaborative software [15]. It consists of multiple columns that represent different stages of a workflow. In the example of software development these columns contain tasks which can be moved to the respective column matching the state they are currently in (e.g. *open*, *in progress*, *done*) [15]. If tasks are assigned to team members, the assignee is visually indicated. The board allows visually conceiving the current state and progress of the development, balancing the workload between team members effectively, minimising the work in progress, and identifying bottlenecks [15]. In recent years many different software solutions offering Kanban boards such as Trello [56] or Atlassian Jira [54] have emerged. Having millions of users [82] these tools are nowadays not only used for software development, but all kinds of fields and are helping teams and individuals manage their projects [6].

In the Kanban method, the Kanban board is the central place to store information about a project and process. However, the effectiveness of the visualisation can still be improved [6]. When taking a closer look at a Kanban board in the context of general project management, it is noticeable that its visualisation capabilities are limited because no nesting of columns is intended. As projects and workflows come in different shapes and complexities, the board might not represent these different layers. Software solutions like Jira added features like Epics or sub-tasks to overcome this problem. However, the variety of features creates complexity and steep learning curves when using these tools [43]. Remov-

ing the limitation of the current board \rightarrow column \rightarrow task hierarchy could be an approach that allows visualising different layers without compromising simplicity.

When it comes to the Kanban principle of measuring and managing the flow, the Kanban board is still lacking the capability of providing quantified information about the progress of a project, which is essential for decision making [10, 5]. While software solutions like Jira offer graphical approaches such as burndown charts, currently, project-controlling and planning are still mostly done using additional tools like spreadsheets [40]. A prediction system could potentially enhance Kanban board-based tools by directly supporting project scheduling [40].

This work aims to improve current Kanban board-based implementations focusing on the visualisation capabilities and quantified information about a project's progress. Nevertheless, simplicity in using the software should be maintained. Therefore, this work presents the concept for a board-based task management software that allows arbitrary nesting of elements and features an intelligent recommendation system that supports users in project scheduling. Based on this concept, a prototype application was developed. The concept and usability of the prototype were then evaluated in a user study.

Software prototype The prototype developed for this work aims to remove a bound in a Kanban board regarding the nesting of elements. As in existing implementations, users should be able to use two types of lists: Horizontal lists named Boards and vertical lists named Lists. Boards and Lists can contain an indefinite number of child elements, either Boards or Lists again or the third element type, Items. All elements can be arranged arbitrarily nested, with Items being the only type of element that can not contain child elements. During the refinement of a project's tasks, it is not unusual that a task formerly represented by an item appears to be rather a list of tasks. Therefore it should be possible to convert all elements to all other types of elements. In the above example, the Item could be converted to a List containing the newly refined sub-tasks.

Besides a title and description, resources of different types can be attached to elements. The types implemented for the prototype are Textual (e.g. a label), Numeric (e.g. a price), a User (e.g. the assignee for the task) and Temporal (e.g. start date, due date, duration). Depending on the type of project, especially Temporal resources are constraints that strongly influence the order in which tasks should be processed. The assumption that users can have a hard time finding this processing order for bigger projects motivated the idea of a symbolic intelligence that proposes a scheduling order based on given constraints. While the prototype focuses on the Temporal constraints, the design of the symbolic intelligence allows it to be easily adjustable to other types of constraints such as priority. Ideally, the intelligent feature should support users in finding the currently important tasks without violating the project's different (Temporal) constraints and estimate if the current planning is realistic and manageable or needs adjustments.

User study In order to evaluate the concept of the software prototype, a user study with 13 participants was conducted. Due to implementing arbitrarily nested lists in Kanban boards being a novel approach, it was not clear how users would rate this approach from a usability point of view, if and how they would use the nesting capabilities and its effect on the visualisation of projects. The first part of the user study focused on these research questions. Users had to create an arrangement representing an example project. Additionally, they had to compare it to three different example arrangements of the same project. Participants created example arrangements with a mean maximum nesting depth of 3.08. Furthermore, the participants overall favoured arrangements of the project with a mean maximum nesting depth of 3.77. The values differed significantly from a standard Kanban Board's nesting depth of 2. The results indicate that unbounded nesting can improve the visualisation of projects.

The second part of the study focused on the intelligent recommendation feature. Participants were confronted with two different example projects and instructed to plan them manually. After each manual planning, they triggered the automatic scheduling. The users' manual planning results were compared to the planning created by the symbolic intelligence. Furthermore, the users' attitudes towards such an intelligent feature were evaluated using qualitative questions. The evaluation shows that while the users' manual planning for a small project with five tasks was close to the optimal planning, it differed significantly from the optimal planning in a project with ten tasks, leading to significantly later finish dates and more missed due dates. Finally, the users described the proposal to be helpful, easy to perceive and did not feel patronised by it. Therefore, the evaluation indicates the potential of the symbolic intelligence to support the scheduling of projects.

Contribution This work describes the concept and implementation of a Kanban board-based web application, featuring unbounded nesting for boards, columns and tasks and a symbolic intelligence proposing an optimal task scheduling order. The novel concepts of the prototype were evaluated in a user study. The evaluation indicates that while some improvements are still required, the new features can improve current Kanban board-based task management implementations in terms of visualising projects and providing quantified information about a project's progress to support scheduling of projects without compromising the simplicity of a small overall feature set.

Overview In the following, chapter 2 gives an overview of related work and existing board-based software solutions. In 3 the concepts of the prototype are explained. Subsequently, in chapter 4 a deeper look into the implementation details of the application is taken. The presentation of the user study and its results in chapter 5, is followed by a discussion of the evaluation results in chapter 6, where also an outlook into future work and an explanation of potential use cases is provided. Finally, in chapter 7 the work is concluded.

CHAPTER 2

Related Work

This chapter supplies relevant background information for this work. Therefore, a brief overview of related work and an analysis of some of the existing tools and solutions are given.

2.1 From Lean to Agile

2.1.1 Lean approach

The management of complex projects such as the development of new products is challenging and faces many uncertainties. Even if the requirements are known in the beginning, they may change over time, and it is difficult to allow for all possible details and errors in the process [38]. Quickly adapting to new developments in a dynamic world of businesses and technologies can bring advantages over competitors. Furthermore, to stay competitive, business processes have to be optimised.

In order to cope with these challenges, the so-called Lean approach has been introduced in the Japanese car manufacturing industry in the late 1940s by the industrial engineer Taiichi Ōno at Toyota [39]. The fundamental principle of Lean thinking was to eliminate waste in order to improve the success of the business [52, 39].

One of the tools invented by Ōno to execute Lean thinking in praxis is Kanban [28]. At Toyota, it acted as a just-in-time scheduling system for the distribution and production of products. In contrary to a push system, where products and their parts are produced based on a preliminary schedule, potentially generating waste and consuming space when they are not needed, Kanban is a pull system [28]. In a pull system, new items are only produced once items in the stock are to be distributed, and parts of a production line are only replenished once the next station in the line has consumed the

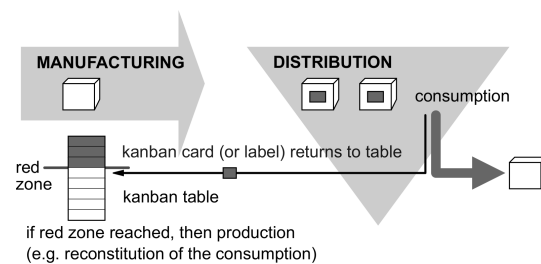


Figure 2.1: Example of a Kanban system for manufacturing and distribution [84].

previously supplied parts [28]. Kanban is a Japanese word for signboard or billboard and describes little physical cards used to implement the pull system. Once a part was consumed by a station in the production line, a card was sent to the previous station, triggering the consumed part's reconstitution. Figure 2.1 shows an example for an implementation of a Kanban system in manufacturing and distribution.

2.1.2 Agile software development

In the last decades, when traditional, inflexible software development methods such as the waterfall method [42] did not meet the reality of the industry anymore, lean thinking and its tools got applied to software development in the form of so-called agile software development (ASD) [26]. In "Lean software development: an agile toolkit", Poppendieck et al. describe seven lean principles and how they can be translated to software development. The principles consist of "eliminate waste", "amplify learning", "decide as late as possible", "deliver as fast as possible", "empower the team", "build integrity in", and "see the whole". In 2001, 17 renowned software developers published the "Manifesto for Agile Software Development" [12], defining philosophies and principles of ASD. Based on these principles, different methodologies and practices such as Scrum, Extreme Programming and Feature-Driven Development [1, 31] got developed.

2.2 Kanban

2.2.1 Kanban method

The ASD methodology Scrum is considered the origin of the evolution of the Kanban method. In the early 2000s, the software team lead David J. Anderson was looking for a systematic way in his ASD team to achieve a sustainable pace of work and minimise the resistance to process changes [9]. These goals motivated him to use a pull system. In 2004 he implemented the first virtual Kanban system for software development at Microsoft [9]. Encouraged by the positive results of this pull system implementation, he developed the Kanban Method in the following years when working with a small team at Corbis. In 2010, he published the results in the book "Kanban: Successful evolutionary change for your technology business" [9]. The principles of the Kanban method are described as "visualise the workflow", "limit work in progress", "measure and manage flow", "make process policies explicit", and "improve collaboratively" [3]. Kanban as a method for managing agile projects and how to apply it in practice is also described by Brechner in 2015 [15].

2.2.2 Kanban board

The core tool to execute the Kanban method's principles in practice is the Kanban board. Based on the original idea, it uses visual cards representing work items and columns representing the stages of the process. The Kanban board can be implemented physically using post-its on a wall or virtually using collaborative software. An example of a Kanban board can be found in figure 2.2. The first column of the board usually contains the backlog of open tasks. Once requiring new work, team members pull new tasks out of the backlog, which

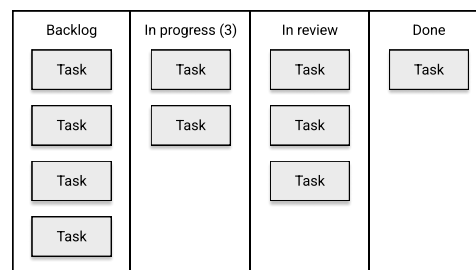


Figure 2.2: Kanban board with a work in progress (WIP) limit for the column "In progress".

leads to a continuous flow and minimises idle times. In order to limit the work in progress (WIP), a maximum number of tasks is defined for certain columns. The visualisation of the workflow allows identifying bottlenecks and measuring the team's performance. The resistance to adjusting the process is low as stages can always be changed. Furthermore, by adding new tasks to the backlog and prioritising it accordingly, new requirements by stakeholders can be fulfilled quickly [15].

2.2.3 Kanban vs. Scrum

Some of the publications presented in the following sections discuss the Kanban method compared to the Scrum method. Also, the Kanban board is mentioned in the context of Scrum. In order to make these comparisons more understandable, a short explanation of the differences between the two methods is given in the following.

Scrum got developed in the early 90s by Sutherland and Schwaber, who also signed the "Manifesto for Agile Software Development" [12], and is, therefore, older than the Kanban method [80]. Considered to be the origin of the Kanban method, it shares most of its principles. However, the process is different.

Scrum is based on short time intervals (for example, two weeks), known as Sprints. At the beginning of the Sprint, the tasks to be accomplished during the Sprint are planned. Within a Sprint, these tasks should not change. The goal of a Sprint is to deliver a functional piece of software. Parts of Scrum are predefined roles of team members like the Scrum master and regular meetings with predefined purposes, like the Sprint review. With the help of these tools, feedback loops are created that help react to customer needs and improve the team's workflow. Table 2.1 shows a comparison of Scrum's and Kanban's key traits.

Originally, a board was not a part of Scrum as described by its inventors [80]. However, commonly, a Scrum board is used to manage the Sprint backlog, which contains the tasks for a Sprint. The Scrum board works in the same way as the Kanban method's board with the difference that the Scrum board gets reset every Sprint, whereas the Kanban method's board does not get reset and therefore contains all tasks of the backlog [25]. While both boards try to limit the work in progress in Scrum, this is done indirectly by limiting the number of tasks per Sprint. In Kanban, on the other hand, this is done directly by limiting the number of tasks in certain columns [25]. Finally, Kanban stresses visually focused information and communication, whereas Scrum emphasises verbal information and communication [10].

	Scrum	Kanban
Cadence	Regular fixed length sprints (i.e., 2 weeks)	Continuous flow
Release methodology	At the end of each sprint	Continuous delivery
Roles	Product owner, scrum master, development team	No required roles
Key metrics	Velocity	Lead time, cycle time, WIP
Change philosophy	Teams should not make changes during the sprint.	Change can happen at any time

Table 2.1: Comparison of Scrum and Kanban [77].

2.2.4 Benefits and adoption in software development

In recent years, multiple researchers studied the benefits of the Kanban method for software development. Besides empirical studies based on experience reports and questionnaires, systematic mapping studies summarising prior work were published. Generally,

reported benefits of the Kanban include increased motivation of team members [3], improved communication within a team and between stakeholders [46], improved customer satisfaction [3, 46] and facilitating a better workflow [2]. The visualisation through the Kanban Board allows a better understanding of the development process [3, 2] and increases knowledge sharing [37].

Also, research has been conducted comparing the Kanban method to other agile methods. In 2015 Lei et al. surveyed 35 employees comparing Scrum and Kanban through six predefined factors relevant for software development [27]. Although both methods led to successful software projects, the results suggest that Kanban has advantages in project scheduling. A reason for this advantage could be Kanban's higher flexibility compared to Scrum, as changes can be made at any point of the development process [10].

While the phase of the adoption is still early, Kanban is seeing a rise in popularity in recent years, and studies indicate a possible shift of agile development teams from Scrum to Kanban [2, 48]. Alaideros et al. even argue in their 2021 paper that the Kanban Method is currently the best among other agile software development methods [6]. Other researchers like Aurisch et al. argue that factors within teams should be considered for deciding whether Scrum, Kanban or even ScrumBan (a mix of both Kanban and Scrum) fits the best for them [10]. Finally, most recent studies conclude that because of the developments being relatively recent, more empirical research has to be conducted to study the adoption in more detail [6, 48].

One inside form of numbers is supplied by the 15th edition of the annual state of agile report published by the company Digital.ai in July 2021 (published by CollabNet VersionOne in previous years) [57]. The results are based on 1,382 completed online survey responses collected from individuals working at companies across a broad range of industries. It reports that 94% of the companies use agile methods, with 86% of the companies using these methods in the area of software development. Kanban practices are used in 61% of the companies, with Kanban boards being the most popular agile planning tool with 77%. However, when being asked which agile methodology the companies follow most closely at the team level, Scrum is first with 66% followed by ScrumBan (a mix of both Kanban and Scrum) with 9%, and Kanban with only 6%. [57]. The numbers indicate that while Scrum, as the older of both methods, is still dominant, many companies nowadays integrate Kanban practices into their Scrum. The popularity of the Kanban board is also confirmed through a study conducted with 14 CEOs of early-stage software startups published in 2019 [49]. Here, the Kanban board was among the most used agile practices.

2.2.5 Kanban in other fields

Originally developed for the manufacturing industry, Kanban and other agile methods were successfully adapted for the application in the software development industry. However, this section shows that the usefulness of Kanban and its board is not limited to that field. According to the 15th state of agile report by Digital.ai, the adoption of agile methodologies, tools and processes by non-IT lines of business doubled since the previous report [57]. The big increase in adoption in the last year is reasoned through the pandemic's impact on work culture. The areas with highest adoption besides software development (86%) and IT (63%) were operations (29%), marketing (17%), security (17%) and human resources (16%) [57].

However, also outside of companies, Kanban can be applied. An excerpt of publications studying other application areas is presented in the following. In the 2011 book "Personal Kanban", Benson et al. explain how to use the Kanban board and method for organising ones personal life [13]. Bass et al. suggested Kanban principles and the Kanban board as

a support for writing academic publications [11]. In 2019, Hidalgo et al. studied the adoption of the agile method Scrum for a distributed research initiative of 17 researchers [23]. In order to visualise the workflows, the web-based project management software Trello was used, which offers virtual Kanban boards. Data was collected through interviews and statistical analysis of the software's activity data. The virtual Kanban board was perceived as practical and aligned with the need to specify, visualise and assign tasks of scientific work. Reportedly, it was easy to get an overview of the work and work assignments [23]. However, some researchers found it hard to adapt to the new methodology. As a result, not everyone contributed to maintaining the Kanban board to the same extent, which led to a sometimes outdated state of the board [23].

Saltz et al. compared different agile methods for the application in data science projects and found the Kanban method to be the most effective one [44].

Fitriawati et al. successfully applied the Kanban method for designing an information system for kindergarten learning evaluation [21].

Santirojanakul et al. applied a virtual Kanban board in the context of sports science and found that it supports overcoming collaboration and communication challenges for sports scientists, executives, staff, and sports associations [47].

Matthies et al. conducted a study with 18 students, in which they applied agile methods to a university capstone course [30]. The subject of the course was to develop a collaborative software project in small teams with the target of learning agile methodology hands on. The course design applied Scrum in the first weeks and then shifted to Kanban in the later phase of the project, once the students were familiar with the Scrum practices. A survey conducted at the end of the course showed overwhelmingly positive students' reactions regarding the integration of Kanban into the course. The analysis of development data collected during the course further indicated the student's successful adoption of the new methodology [30].

2.2.6 Challenges and areas of improvement

As seen in the previous chapters applying the Kanban method to manage projects in different fields can have substantial benefits [2]. However, also problems and challenges were identified by researchers.

First of all, while being relatively simple, the Kanban method does not guarantee success [3, 2]. Regardless, its simplicity makes it efficient when being combined by other tools [53].

Furthermore, compared to Scrum, which has a well-documented process framework, no such framework is currently available for Kanban [2]. Additionally, due to being relatively new, training and specialised skills are still lacking [4]. These factors lead to adoption problems, as already seen in the previous chapter [3, 2]. A key factor in overcoming these adoption problems can be an increased application and education at academic institutions [6].

Other criticism concerns the Kanban board as the central place to store all information about the process. The mentioned lack of a supporting framework and proper training can lead to the Kanban board being overcomplicated by the method's practitioners [10]. Furthermore, while one advantage of the Kanban method is limiting the work in progress in certain stages directly, it is hard to determine the optimal number for these WIP limits, which is crucial in order to avoid schedule delays [5, 18]. Additionally, while quantified information about a project's progress is essential for decision making [18], the Kanban board has problems in providing this information [10, 6]. Therefore integrations with other methods such as Earned Value Analysis (EVA), a project management technique for measuring project performance and progress, got proposed [36]. Other commonly used graphical approaches like cumulative flow diagrams provide visualisation but lack quantitative information [5]. Alaideros et al. conclude that features of progress tracking in a Kanban

board should consist of real-time updates, showing schedule deviations, providing quantified information and a progress status [5]. Research should be conducted in order to find approaches to improve the visualisation of the Kanban board to that regard [6].

2.3 Virtual Kanban board implementations

This work focuses on the implementation of a virtual Kanban board. Therefore, this section gives a brief overview of existing implementations for scientific publications and consumer products.

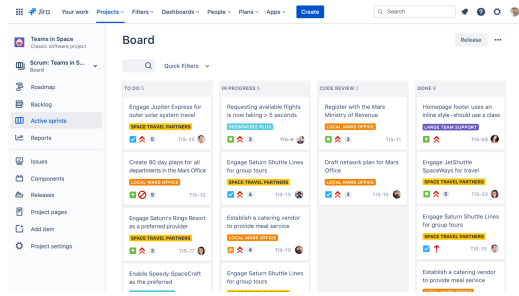
2.3.1 Prototype implementations

Nakazawa et al. developed and evaluated multiple virtual Kanban board prototypes to improve existing implementations. One of their publications studied a board prototype featuring horizontal separated units to show tasks assigned to each developer within a team. In a controlled experiment conducted with ten students, the prototype was compared to existing tools [33]. The results indicate that dividing a board into meaningful horizontal units can allow perceiving relevant information faster. Building upon the previous prototype, further work consisted of a prototype that, besides the horizontally separated units, additionally allowed configuring and setting work in process limits for each of the developers [34]. The prototype was used to manage a student project with ten developers and evaluated via questionnaires. While generally the tool was rated positively in terms of visualisation, reportedly the functionality to limit the work in process per developer led to improved task assignment [34]. In another work, Nakazawa et al. developed a prototype that tried to combine the advantages of a physical Kanban board as a place to gather and discuss project details, with the advantages of a digital Kanban board [32]. It featured a virtual Kanban board shown on a large screen and allowed modifying the state of the board using smartphone gestures. Saltz et al. also developed a Kanban board prototype that tried to improve the understanding of the individual contribution of a team member by visualising the work done per team member per week [45]. This visualisation was perceived as helpful and easy to understand in a case study. Finally, in order better visualise the progress of a project, Alaideros et al. developed a prototype tool for progress monitoring [7].

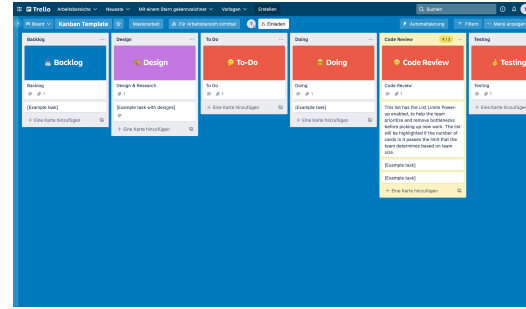
2.3.2 Consumer products

Many companies offer virtual Kanban board implementations for different use cases and extend them with various features. The following describes four of these tools in more detail. All presented applications are web-based; Atlassian Jira, Trello and Asana also provide mobile applications.

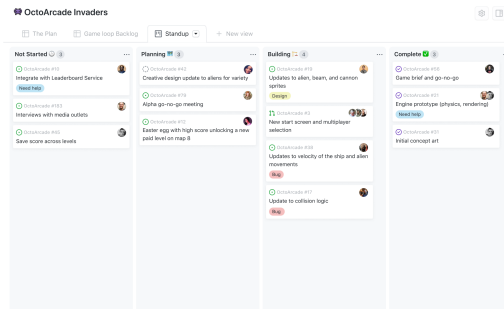
Atlassian Jira Atlassian Jira [54] was the most recommended agile software in the 15th state of agile report with 81% [57]. Jira is targeted at companies and businesses that already perform or want to adopt agile practices. The agile methods Scrum and Kanban are actively supported through features. Scrum boards, Kanban boards (with the option to limit work in progress), road mapping and different visualisation and reporting capabilities like Sprint reports or burndown charts are offered. Jira is feature-heavy, but the variety of features and concepts make it potentially hard to learn and adopt [43]. A screenshot of an example board can be seen in figure 2.3a. The typical board → column → task hierarchy is extended through concepts like Epics allowing the combination of multiple tasks into a body of work and sub-tasks allowing the split-up of tasks into smaller portions of work. Sub-tasks can be viewed directly on the board via a sub-task view.



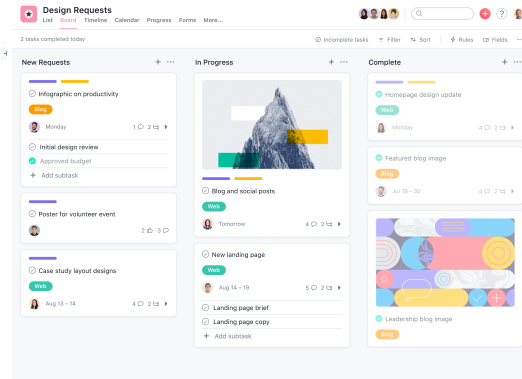
(a) Atlassian Jira [54].



(b) Trello [56].



(c) GitHub Projects [70].



(d) Asana [62].

Figure 2.3: Virtual Kanban board implementations.

Trello Being first released in 2011 Trello [56, 24] got acquired by Atlassian in 2017 [88]. It is one of the most popular implementations of Kanban boards, having self-reported 50 million users in 2019 [82]. Trello targets individuals and teams that want to use the advantages of a Kanban board for organising and planning their projects. Figure 2.3b shows a Kanban board in Trello. As Trello focuses on the board as its main tool, it is more lightweight and easier to learn than Jira. Nevertheless, recently added new features consist of automation and different views like a calendar or assignment view. To conform with the Kanban method, the number of tasks per column can be limited. Trello offers the typical board \rightarrow column \rightarrow task hierarchy; however, checklists can be added to tasks to represent sub-tasks. Furthermore, different resources like dates, images or tags can be attached to tasks.

Asana Originally released in 2012 [85], Asana [62] now has more than 30 million registered users according to their 2021 Form 10-K report [61]. Asana is a general work management platform targeted at teams to organise, track, and manage their work. For this purpose, it offers many different features and concepts. Besides timelines, calendars, reports, forms and automation, it also features Kanban boards. While not being directly designed for agile methods, Asana claims to support Scrum and Kanban. However, as of January 2022, no task limits per column can be set. Like Jira and Trello, Asana extends the typical board \rightarrow column \rightarrow task hierarchy by offering sub-tasks that can be added to tasks. Figure 2.3d shows an example board.

GitHub Projects GitHub Projects [70] is part of GitHub [69], a source code hosting service based on the version control system git. GitHub got acquired by Microsoft in 2018 and is the largest source code hosting service as of November 2021 [86]. It targets software

developers who want to manage their software projects in the same place as their code repositories. GitHub Projects allows managing the project related issues in a Kanban board, as seen in figure 2.3c, with the typical board \rightarrow column \rightarrow task hierarchy. Alternatively, in the recently released new version of GitHub Projects, a board can also be viewed in a table view. Furthermore, it features rule-based automation to transfer tasks to certain stages on certain events automatically. Overall it is considerably more lightweight than its competitors mentioned above.

2.3.3 Improvement of existing tools

Prior studies examined the usage of existing virtual Kanban board implementations in practice. Some of these studies are discussed in the following. The findings mostly correspond to the areas of improvement reported in section 2.2.6. Saad et al. conducted a focus group study with ten software developers examining the developer's experience (DX) of virtual Kanban board tools [43]. As the main issue, scope understanding was identified, which describes difficulties in understanding the tasks and vision of a project due to the lack of features within the tool to visualise information and project steps. It was followed by project needs, which describe the lack of features that could support developers in their daily work. The third most reported issue regarded complexity, which describes problems in the usage of the tools due to being too complex and too hard to learn [43]. Another study was conducted by Raith et al., who conducted semi-structured interviews with five agile coaches working at five different projects with a total of 30 Scrum and five Kanban teams [40]. Among the reported potentials for improvement was a more detailed visualisation of project progress and forecasting to support project-controlling, which is currently mostly done using additional tools like spreadsheet applications. The forecast could consist of a prediction of whether a project can be done in time or not [40].

2.4 Data-driven estimation of agile software projects

As seen in the previous section, one way of improving Kanban boards is providing quantified information concerning a project's progress. As part of this, estimating a project based on its progress could be valuable. Therefore, a brief overview of data-driven estimation techniques in agile software projects is given in this section.

Estimating the effort and duration of a software project or certain tasks within the project is a key factor in software management [50]. Unrealistic estimates threaten the project's success, and inaccurate estimates can lead to misallocation of resources [8]. One way to obtain valuable project estimation is consulting experts. However, in recent years, research has been conducted to find data-driven effort estimation techniques based on historical data of a project [50, 8]

2.4.1 Effort estimation

The most frequently used metric for describing the effort in software development are story points [8]. Story points are numeric values attached to user stories or tasks in the backlog and describe the relative volume of work for that item [29]. The leading technique practised by Scrum and Kanban teams to determine the number of story points for a task is known as planning poker [29]. Based on the story points, a team's velocity can be calculated, specifying the number of story points completed per time unit [20]. However, factors like the experience of team members and external pressure limit the quality of the story point

estimates [8]. Hence, one direction of research focuses on finding methods to automatically estimate the effort of user stories or tasks within an agile project. Different techniques and algorithms like decision trees, Bayesian networks, support vector machines or neural networks have been studied [8]. One example is Choetkiertikul et al., who proposed a deep learning model based on long short-term memory and recurrent highway network to estimate story points [17]. The model utilises the title and description of a user story as input. Besides the textual definition of a task, a systematic literature review by Alsaadi et al. identified four independent characteristics previously been used as data sources for the estimation of user stories [8]. These consisted of personal factors, like experience and developer skills, product factors, like priority and complexity, process factors concerning specification quality and estimation factors, such as story points estimated through planning poker.

2.4.2 Lead time estimation

Another research direction focuses on determining the actual lead time based on the tasks' or user stories' effort. The lead time is defined as the time elapsed between adding a task to the backlog and its completion. Story points and velocity are not the only factors influencing a task's lead time. When looking at the Kanban method, the lead time is also impacted by other factors like the task's priority and the frequency of backlog changes [51]. One approach to account for all these factors is an influence diagram, as proposed by Welfen et al. in 2022 [51]. It consists of a Bayesian Network with nodes representing the lead time factors. Initially, the nodes in the network have to be weighted, which was done through the collection of data of a Kanban team [51]. Alternatively, parts of the data could be substituted using approaches described in section 2.4.1. The influence diagram yields probabilities for certain lead times as an output. The researchers concluded that these probabilities could help Kanban teams in communicating to stakeholders and support decision making [51].

This work aimed to implement and evaluate a Kanban board-based task management application prototype with unbounded nesting capabilities and an intelligent recommendation feature. Building on the findings of chapter 2, this chapter describes the concept of the prototype and explains relevant design decisions.

3.1 Goals

1. Everything is a list As seen in chapter 2, Kanban board-based task management applications generally make three elements available to the user: Boards, Lists and Items. These elements are the base components that help users organise and visualise their tasks and workflow and should therefore be available in the application. However, while distinguishing between those elements is important for the user, this distinction is not necessary from a technical point of view. Intuitively, a Board is just a list of child elements arranged horizontally. In a List, child elements are arranged vertically. Moreover, an Item is a list that does not contain child elements. In order to display the elements correctly to the user,

- a list containing nothing can be displayed as an Item.
- a list containing only Items can be displayed as a List.
- a list containing Lists (which contain Items) can be displayed as a Board.

Therefore, to model the prototype application, the goal was to abstract the three elements into a single entity, a list.

2. Unbounded nesting As seen in the previous chapter, the Kanban method utilises the Kanban board as its main tool to visualise a project. However, related work also identified room for improvement regarding this visualisation. Notably, all existing implementations of a Kanban board consist of the typical Board \rightarrow List \rightarrow Item hierarchy. This hierarchy limits the expressiveness and flexibility of the visualisation capabilities for different scopes, levels of detail and hierarchies within projects. Therefore, the second goal was to remove this bound of a Kanban board to improve its visualisation capabilities. In the prototype,

it should be possible that Boards and Lists can contain Boards, Lists and Items as sub-elements. Furthermore, it should be possible to arrange them to an unlimited depth. Removing these bounds is a novel approach; therefore, user experience and design challenges are expected. Some of the challenges for usability with unbounded nesting are examined in the following.

3. Intelligent (scheduling) recommendations As a third goal for the application, quantified information about a project's progress should be provided through intelligent recommendations to support users in project scheduling. Furthermore, it might be beneficial if the recommendations are self-explanatory and unobtrusive. Motivated by the findings of chapter 2, the idea for a symbolic intelligence came from the assumption that for humans, it is hard to plan with different (temporal) constraints in complex projects. While it is easy to set temporal goals within projects and guess how long a single task will take, it is hard to estimate when the project will finish. Also, sometimes it might be possible to reach an earlier finish date if the order of the tasks is optimised. Therefore, the intelligent scheduling feature proposes a task order that minimises the overall project timespan and the number of missed due dates. This recommendation can be used to evaluate the current project plan.

3.2 Ideation and development process

This section gives a brief overview of the prototype's ideation and development process. Based on the first two goals explained in the previous section, an initial prototype consisting of an unbounded nested structure and the base features was built. After that, during the ideation process for the implementation of the recommendation feature, existing boards' data from GitHub Projects and Trello was fetched via publicly available APIs [55, 71]. For potentially using this real-life data for the intelligent recommendation feature's evaluation, logic was implemented to adapt this data into the application's data model. Also, the fetched data was plotted in different ways to find a direction for the intelligent recommendation feature. The visualisation provided interesting insights and generated ideas, which could be used for a recommendation feature system based on statistics or machine learning. Some of these ideas are discussed in more detail in chapter 6. However, a more practical approach independent of large data sets was finally chosen for the prototype. The approach focuses on a symbolic intelligence based on a constraint solver to find the best order of tasks and evaluate the current temporal planning. As a preparation for the symbolic intelligence's implementation, the resource concept was added, which provided the basis for the constraints used by the symbolic intelligence.

3.3 Features of the prototype

The following provides a deeper look into the prototype's features.

3.3.1 User and workspace

While being very important for a consumer product, registering, managing and authenticating different users was not the focus of this prototype. However, a simple page is available where different users can be created. These "users" provide separate workspaces that were used to present the different tasks of the study (see chapter 5). Selecting one of these users opens the main application featuring a user's workspace. When first selecting a user, an empty workspace is shown. New elements created inside the workspace are arranged vertically below each other. The workspace can be considered the root element, which can

not be changed but acts the same way as a List or Board element. Its child elements can be repositioned, and further down elements can be moved up to the root.

3.3.2 Elements

In the prototype version evaluated in the user study, each element type is marked with a different colour (blue for Board, red for Lists and white for Items). An elements type is also indicated in textual form at the top of the element. Each element has a title and an optional description. For Item elements, resources can be configured (see section 3.3.3). An arrangement for a standard Kanban board is depicted in figure 3.1.

In order to perform relevant actions for the element, multiple buttons on the right upper corner of an element are available. Icons and tooltips indicate the action type. The actions include adding new child elements, modifying the element, converting the element, moving the element and deleting it.

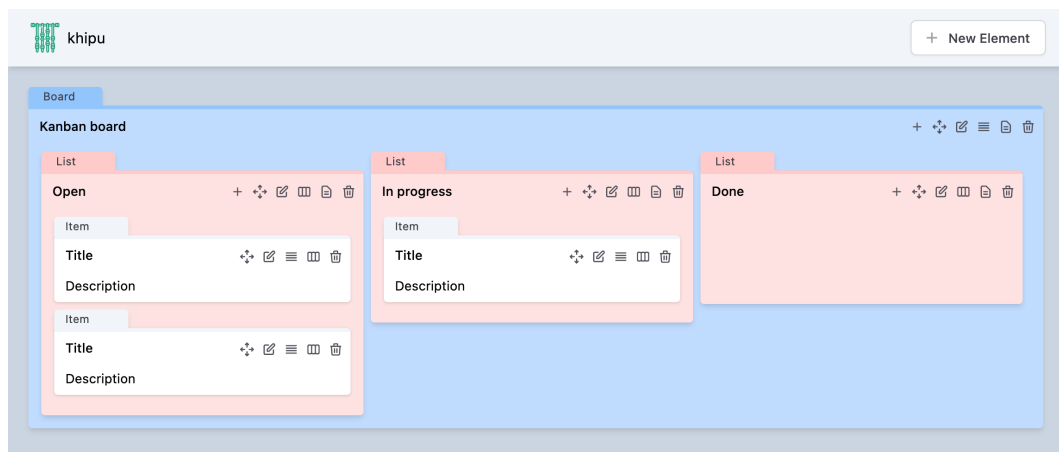


Figure 3.1: View of a workspace featuring the arrangement of a standard Kanban board.

Nested arrangement When nesting multiple elements, parent elements automatically adapt their size to fit larger child elements. When elements grow bigger than the screen width, they get scrollable. An example of an arrangement with unbounded nesting can be found in figure 3.2. In existing implementations, elements are often coded with similar colours like white and light grey. However, when experimenting with this in the development phase of the prototype, the UI seemed to get cluttered for deeper nesting situations, and elements could hardly be distinguished. Therefore, the elements were colour-coded in widely different colours to distinguish them even in deeper nesting situations.

Creating elements New elements can be added to every Board and List element or the root of the application by clicking the plus icon (+). Figure 3.3a shows the modal, where the element's details like the desired type, title, and description must be entered. If the new element is a Board or a List, the user can create child elements for the new element in the modal.

Modifying elements The pen icon (✎) allows editing the details of an element. Figure 3.3b shows the corresponding modal, where the element's updated details must be entered.

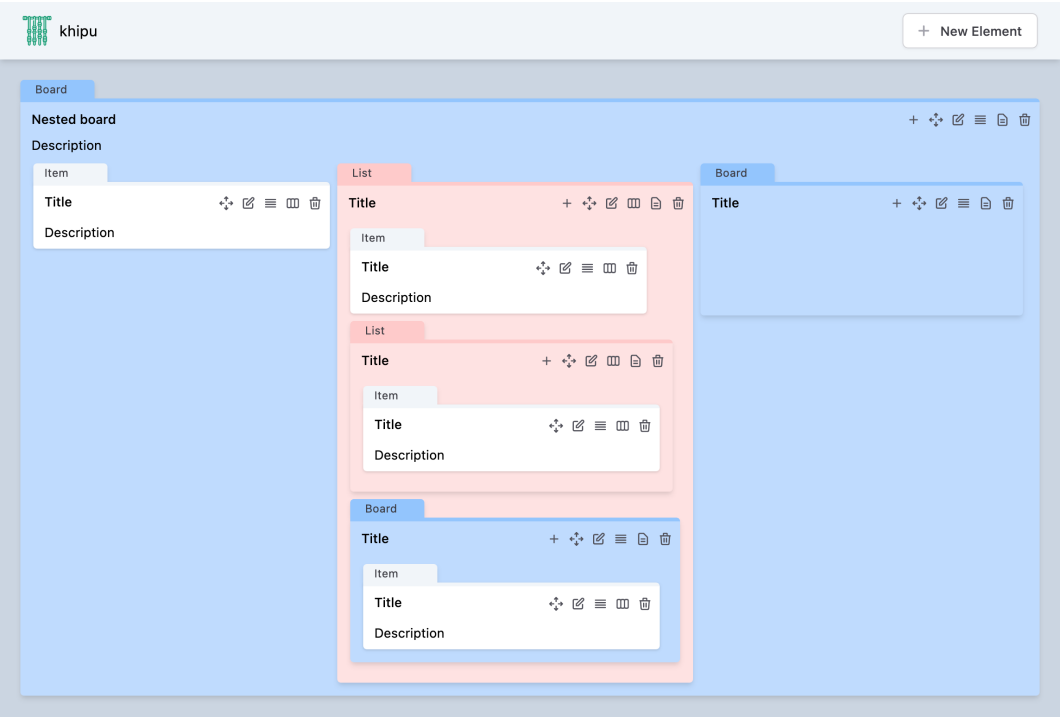
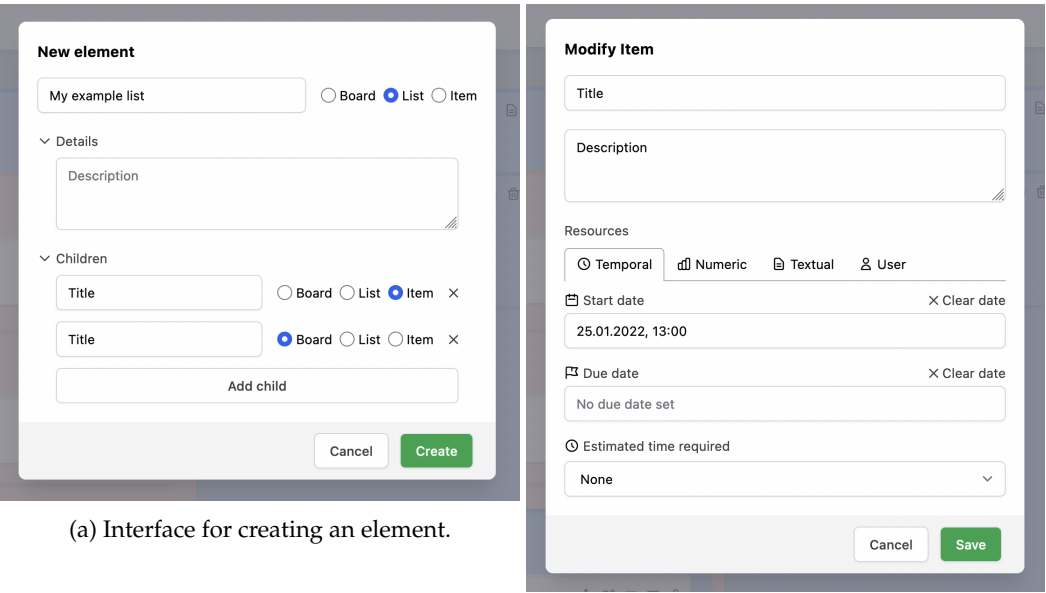



Figure 3.2: View of a workspace featuring an arrangement with unbounded nesting.



(a) Interface for creating an element.

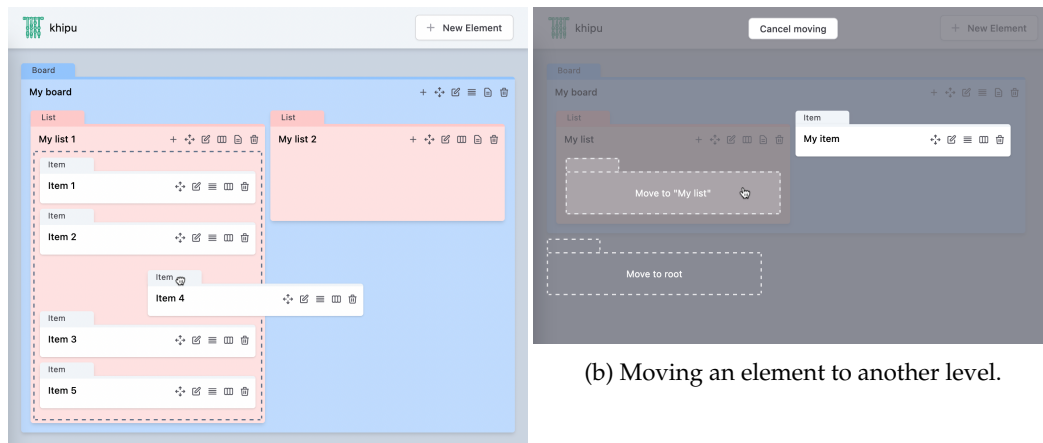
(b) Interface for modifying an element.

Figure 3.3: Interfaces for creating and modifying elements.

Deleting elements The bin icon () allows deleting an element. As this is a destructive action that can not be undone, it must be confirmed by the user. If the element contains child elements, these will be deleted as well.

Repositioning elements The application allows repositioning elements within their parent using drag and drop as seen in figure 3.4a. Drag and drop should be considered an important feature, being an intuitive and natural way of ordering elements and gives the user a feeling of control by simulating elements as real-life objects. It is also implemented for existing Kanban board-based implementations as described in section 2.3.2.

Moving elements to new parents: When it comes to moving elements from one parent element to another (for example, an Item from one List to another List), in existing implementations described in section 2.3.2, this likewise is done using drag and drop, like repositioning. However, Drag and Drop presents a challenge when dragging an element from one level to another in a deeper nested scenario. While experimenting with this, as different target areas overlap, it was hard to determine the correct target for the Drop, not only for the user but also from a technical point of view. Therefore, element locations resulting from a Drag and Drop interaction often did not match the user's expectation. Finally, it was decided to limit Drag and Drop capabilities such that elements can only be dragged to parents on the same level. Moving elements to a different level in the application is implemented through a "move" button (\leftrightarrow). As seen in figure 3.4b after clicking it, all potential locations for the element are highlighted, and the users can select where they want to move the element. Evaluating this approach was an important part of the user study and is further discussed in chapter 5.



(a) Repositioning an element using Drag and Drop.

(b) Moving an element to another level.

Figure 3.4: Drag and Drop interaction and moving of elements.

Converting elements Generally, it is natural that the true scope is often unknown when planning a task or project. When working on it, small tasks can rather turn out to be a list of multiple tasks or vice versa. In a task managing application, this would require restructuring. As a goal of the concept was to remove limitations, the user should be supported in performing this kind of restructuring. Therefore, an element can be converted to any other type of element. The conversion can be done by clicking the respective button marking the desired element type (\square , \equiv , \square). Conversions of elements result in the following:

- Board \leftrightarrow List: View changes from horizontal to vertical and vice versa, colour changes, all children are preserved.

- **Board/List → Item:** The titles of direct child elements are preserved as bullet points (marked with $-$) written into the description of the new item. As deeper sub-elements will be lost with this conversion, it has to be confirmed by the user.
- **Item → Board/List:** If the description of the items contains bullet points marked with $-$, these will be added to the converted element as child Items.

3.3.3 Resources

The Item element can have more properties in existing implementations described in section 2.3.2 than just a title and a description. For example, labels can be added, a user assigned, or a date configured. These properties were generalised as so-called resources. Four types of resources applicable for various use cases were implemented. The types are explained in the following. Resources can be configured during the creation and modification of Items as seen in figure 3.3b. Figure 3.5 shows an example for an Item with all available types of resources configured.

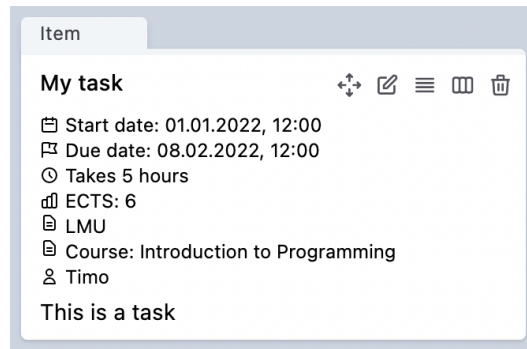


Figure 3.5: Item element with the temporal resources start date, due date and duration, one numeric resource, two textual resources and a user resource.

- **Numeric:** Consists of a text label and a number value.
- **Textual:** Consists of a text label and an optional text value.
- **User:** Consists of one of the existing users (see 3.3.1), which can be selected from a dropdown menu.
- **Temporal:** Consists of three optional fields:
 - *Start date:* Defines the date and time at which a task can be started at the earliest. Can be entered or selected using a date picker.
 - *Due date:* Defines the date and time at which a task is due. Can be entered or selected using a date picker.
 - *Duration:* Defines the expected duration for the task. The selection is implemented using a dropdown menu with predefined values between 5 minutes and 10 hours. Predefined values are sufficient for estimation and easier to enter. The maximum value was set to 10 hours to encourage splitting longer tasks into sub-tasks.

3.3.4 Events

Recording a history of a user's actions was required to enable certain calculations within the intelligent recommendation feature, like determining a task's time in progress. Therefore, every time an action is performed, it is written into an event log. This approach is similar to existing implementations like Trello or GitHub Projects, as the structure of their public APIs indicates.

3.3.5 Recommendations: Symbolic intelligence

A symbolic intelligence providing recommendations is implemented to support users in scheduling their projects. Its purpose is to find the optimal order of tasks within the project and evaluate due dates by comparing them to the calculated finish dates. The three temporal resources defined in 3.3.3 were used as constraints for this planning. The scheduling result consists of a recommendation for a task execution order and an estimation for the project's finish date.

The scheduling can be triggered by clicking the clock icon (🕒) on any Board in the workspace that has at least two sub-Lists or Boards. As explained in more detail in section 4.2.4.2 this limitation was added in order to determine which tasks are relevant for the scheduling. Then, the best order for the tasks within the Board is calculated, which takes about five seconds. The spinning clock indicates that the calculation is still ongoing. The result of a scheduling calculation can be seen in figure 3.6. Every task that is considered to be *open* or *in progress* is marked with an indicator showing its optimal position in the execution order. A green colour indicates that the due date can be reached using the proposed order. A red colour indicates that the due date can not be reached. Sub-tasks are considered in an aggregated form (explained in more detail in section 4.2.4.2). The Board also gets indicated with a field showing the total remaining duration for the project. Hovering over any indicator reveals further information like the calculated start date, finish date, and remaining duration. Furthermore, besides the projects start and finish date, the tooltip for the Board also reveals the work schedule used for the calculation. The work schedule is not configurable directly in the UI in the prototype but can be configured in the database. It is described in more detail in section 4.2.4.2.

3.4 Design and user experience

The design and user experience of the prototype are oriented at existing implementations described in section 2.3.2. While not being the main focus of this work, an iterative design process loosely following the Double Diamond model [35] was applied by showing different evolution stages of the prototype to a small group of people. Their feedback influenced design decisions such as the colour selection, tooltips, moving of elements, and the presentation of the recommendations. The evaluation of the prototype's usability is described in chapter 5.



Figure 3.6: Scheduling result for the Board "Tasks of the week".

This chapter describes the implementation details of the prototype based on the concept presented in the previous chapter.

4.1 Architectural Overview

Figure 4.1 presents an overview of the application's architecture. A server-client architecture was chosen with a frontend providing a user interface and a backend holding the business logic and performing relevant computations. The parts of the application are described in more detail in the following.

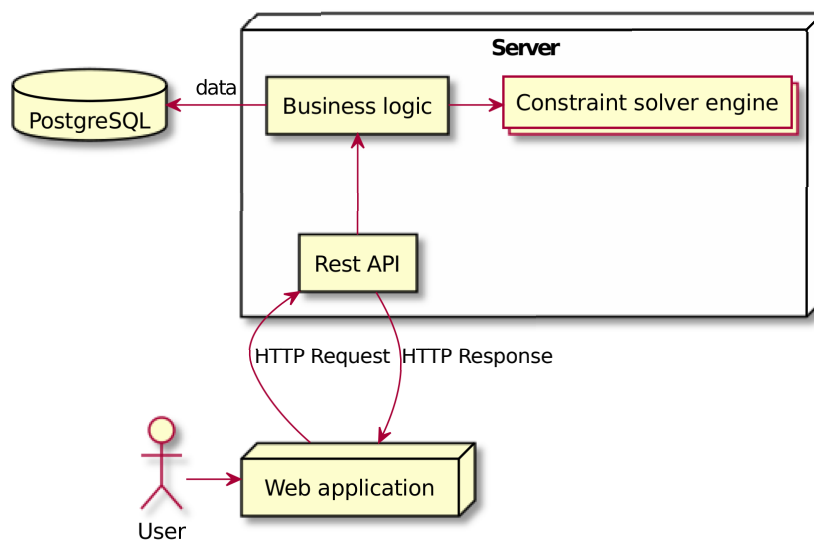


Figure 4.1: Architecture of the prototype.

4.2 Server

4.2.1 Technology stack

The application's backend is written in the Scala programming language [76], which is a strongly typed multi-paradigm language that compiles to Java byte code and runs on the Java virtual machine. It allows object-oriented as well as functional programming. The backend was designed as a RESTful API, with the advantage that different frontends, like a web application or mobile applications, can communicate with it via HTTP requests. In order to build the API, the Scala library Akka HTTP [63] was utilised, which provides a toolkit for building HTTP servers. The backend stores all application-related data in a PostgreSQL database [59]. In order to read from and write to this database, the Scala library Slick [78] was used, which allows writing SQL-Queries as typed Scala code. As Scala compiles to Java byte code, it also allows accessing Java libraries. Therefore the Java library OptaPlanner [60] was utilised to build the constraint solver for the symbolic intelligence.

4.2.2 Data model

As seen in chapter 3 the state of the application consists of different elements called Board, List and Item, which can be arranged in an arbitrarily nested way. This state can be seen as a hierarchical tree with nodes representing elements and vertices representing nesting relations. The tree was modelled in the backend using an adjacency list data structure. The entire class diagram can be seen in figure 4.2 A node is represented by the class `WorkflowList` with a property `parentId` that references its parent node, being the element where the workflow list is nested inside. If the workflow list is not nested inside another element, the `parentId` equals null. Furthermore, the property `listType` specifies whether the workflow list should be displayed as a Board, List or Item to the user. The property `position` stores the order in which children are arranged inside a parent. It is updated once the position changes. The adjacency list structure allows updating individual nodes without changing the whole tree. Furthermore, it allows storing the state in the relational PostgreSQL database.

Because of the technologies used (see section 4.3), it was more practical to handle the full tree rather than individual nodes in the frontend. Therefore, when the client fetches the state, the backend builds the full tree and returns it as nested lists.

4.2.3 Rest API

The backend offers HTTP endpoints that allow clients to retrieve and manipulate the application's state and trigger the symbolic intelligence. A list of all relevant endpoints can be found in table 4.1. The payloads and responses of endpoints are transferred using the JSON data format. Because the application was built as a proof of concept, no proper authentication is performed. However, all requests to `/workflowlist` require the user's UUID as an Authorisation header and return the HTTP error `401 Unauthorised` if the UUID is not valid or the user is not marked as active.

4.2.4 Symbolic intelligence

4.2.4.1 Constraint solver

The goal of the task scheduling algorithm consists of finding the optimal order of a list of tasks such that firstly the number of missed due dates, and secondly, the final task's finish date is minimised. This problem is called the scheduling problem in the following. Solving

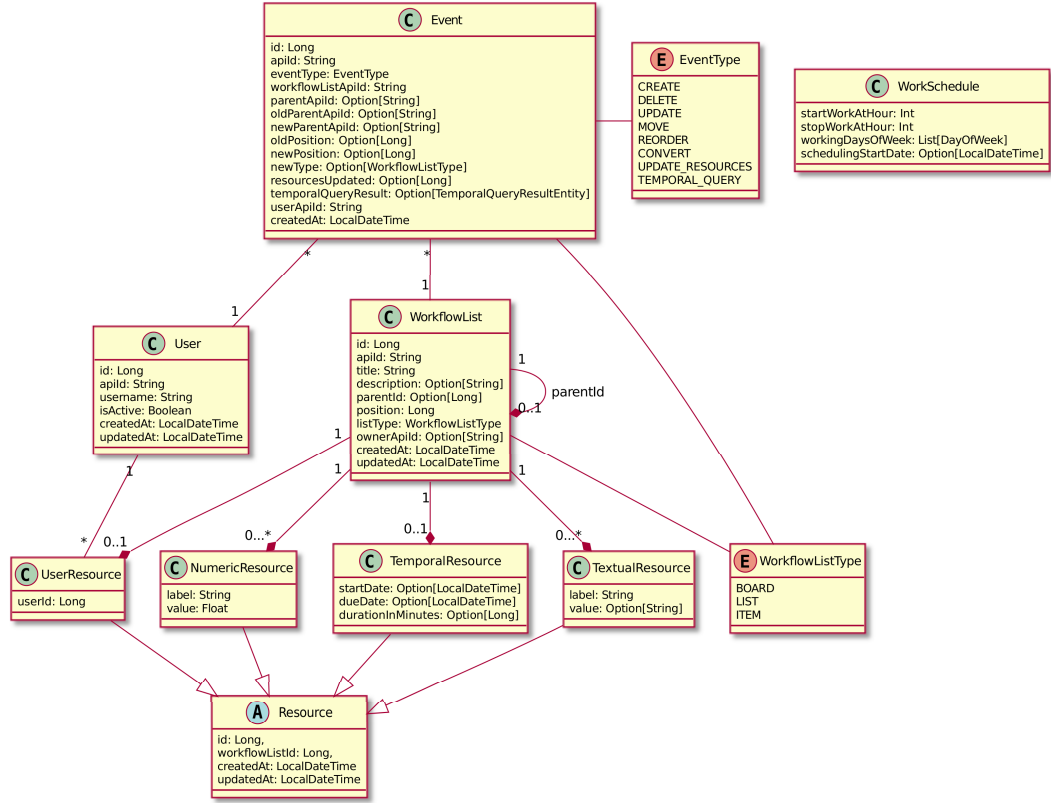


Figure 4.2: UML class diagram of the backend.

Method	URL	Description
GET	/user	Retrieve all users.
GET	/user/{uuid}	Retrieve a single user by given UUID.
POST	/user	Create a new user.
GET	/workflowlist	Retrieve all workflow lists.
POST	/workflowlist	Create a new workflow list.
DELETE	/workflowlist	Delete a workflow list.
PUT	/workflowlist/{uuid}	Update a workflow list.
PUT	/workflowlist/{uuid}/resource	Update the resources of a workflow list.
PUT	/workflowlist/{uuid}/type	Convert a workflow list to a new type.
PUT	/workflowlist/{uuid}/position	Change the position of a workflow list within its parent.
PUT	/workflowlist/{uuid}/parent	Move a workflow list to a new parent.
GET	/workflowlist/{uuid}/scheduling	Retrieve a scheduling proposal from the symbolic intelligence.

Table 4.1: HTTP endpoints of the backend.

the scheduling problem in reasonable runtime showed to be no trivial task. Initially, the problem was solved using a brute-force algorithm. Here, the number of due dates missed and the total finish date is calculated for every possible order of tasks. The best solution is obtained by sorting all evaluation results by due dates missed and total finish date ascend-

chained planning variable pattern is applied for the scheduling problem by using the previous task as a planning variable. The chain then represents the order of the tasks. However, no previous task is available for the first task of the order. As nullable planning variables are not supported for chained planning variables, the abstract class `TaskOrAssignee` is used as a planning fact instead of the `Task` class. The classes `Task` and `Assignee` inherit from the abstract class. For the first task in the chain, the `previousTaskOrAssignee` is represented by an `Assignee` marking the chain's anchor. The abstract class `TaskOrAssignee` contains a so-called inverse shadow variable `nextTask`, which references the task holding the `TaskOrAssignee` as a `previousTaskOrAssignee`. Shadow variables are updated every time a planning entity changes. Two more shadow variables are used to represent the time at which a task gets started in the current planning (`startedAt`) and the time at which a task gets finished (`finishedAt`). The shadow variable `startedAt` is calculated as the most recent time out of the start date of the current task and the finish date of the previous task. The shadow variable `finishDate` is calculated as `startedAt` plus the task's duration. Another important part is the planning solution, annotated by `@PlanningSolution`. It wraps all planning entities used for planning and all assignees (in the scheduling problem case, always just one assignee) into a single data set. The `TaskSchedule` class represents the planning solution for the scheduling problem. The list of planning entities (here, list of tasks) is annotated by `@PlanningEntityCollectionProperty`. As the assignees do not change during planning (they are always just the chain's anchor), they are annotated as problem facts by `@PlanningFactCollectionProperty`. As a third defining property for the planning solution, a score annotated by `@PlanningScore` has to be defined. The score is calculated during the planning and used to compare different planning solutions. It is described in more detail in the following.

Constraints and score In addition to the model for the planning problem, constraints have to be defined. The OptaPlanner engine calculates the score of a planning solution needed to determine the best solution based on these constraints. An overview of the constraints used for the scheduling problem can be found in table 4.2. Three constraints levels are used for the scheduling problem: hard, medium, and soft constraints. Hard constraints are constraints that must not be broken; otherwise, the solution will not be feasible. Medium and soft constraints are constraints that should ideally not be broken. The `HardMediumSoftScore` class was used to represent the score of a solution, which results in score-values such as `-1hard/-1medium/-100soft`. A hard score of < 0 signifies an infeasible solution. The higher the medium and soft scores are, the better the solution meets its constraints. A solution with a higher medium score is always preferred over a lower medium score, regardless of the soft-score value. For the scheduling problem, the start date is used as a hard constraint to ensure the definition of the start date as described in section 3.3.3; it is not possible to start a task before its start date. For each task in the solution started before its start date, the hard score is penalised (decremented) by 1. Furthermore, the number of missed due dates should be minimised for the scheduling problem. This requirement is implemented by using a medium constraint that penalises the medium score by 1 for every task in the planning solution with a finish date after the task's due date.

The second metric indicating a good order of tasks is a minimised project finish date. It is implemented through a soft constraint penalising the soft score with the squared timespan of the project in minutes. The span was calculated as the time difference between the starting date of the calculation (considered the project's starting date) to the finish date of the

final task in minutes. The square of the timespan was used to give this constraint greater weight than the other soft constraints described in the following. Besides the two main features that define a good task order, fine-tuning constraints are added, which comes into play when two solutions are otherwise rated with an identically score. One part of this fine-tuning concerns tasks that are already in progress. It makes sense that these should be finished before starting new tasks. The constraint is applied similarly to the overall finish date constraint. For every task *in progress*, the soft score is penalised with the timespan in minutes until the task's completion. Thereby, solutions that place these tasks earlier in the order get higher scores. Likewise, a constraint is added to ensure that tasks that can not be finished within their due date still finish as close to the due date as possible. Without such constraint, tasks can also be moved to the end of the execution. However, it is arguably better to finish a task with a one-hour than a multi-hour delay, if possible. Therefore, for every task with a missed due date, the soft score is penalised with the timespan in minutes between the due date and the task's finish date.

Constraint type	Goal	Impact on score
hard	Tasks can not be started before their start date.	$\text{HardScore} - 1$ for every task where $\text{task.startedAt} < \text{task.startDate}$
medium	Minimise number of missed due dates.	$\text{MediumScore} - 1$ for every task where $\text{task.finishedAt} > \text{task.dueDate}$
soft	Minimise project timespan	$\text{SoftScore} - (\Delta(\text{startDate}, \text{project.finishDate}))^2$
soft	Prioritise tasks <i>in progress</i> .	$\text{SoftScore} - (\Delta(\text{startDate}, \text{task.finishDate}))$ for every task where $\text{column} = \text{inProgress}$
soft	Minimise exceedance time of missed due dates.	$\text{SoftScore} - (\Delta(\text{task.dueDate}, \text{task.finishDate}))$ for every task where $\text{task.finishedAt} > \text{task.dueDate}$

Table 4.2: Constraints used for the constraint solver.

Configuration The configuration of the OptaPlanner engine requires setting a time limit for the search for solutions. This time was set to 5 seconds, a reasonable waiting time for the user, which resulted in very good solutions during testing.

4.2.4.2 Data preparation, assumptions and calculations

In order to create a list of tasks used as a scheduling problem, some assumptions and transformations have to be made.

Definition of open, done and tasks in progress The scheduling possibility was limited to workflow lists with at least two sub-elements being Boards or Lists to determine the stages *open*, *in progress* and *done*. The first sub-element is always considered as containing tasks that are *open*, the last sub-element is considered as containing tasks that are *done*. If the workflow list for which the scheduling gets triggered contains three or more sub-elements, all elements between the first and the last elements are considered as elements that are holding tasks *in progress*.

Aggregation of subtasks Elements are converted to `Task` entities in preparation for the constraint solving. Considered for the scheduling problem are all elements that are *open* and *in progress*. However, this might also include nested elements. As the constraint solver implementation works with a flat list of tasks, nested elements are aggregated into a single task entity. The sum of all durations is used as the new duration, the latest of all start dates as the new start date and the earliest of all due dates as the new due date for the task.

Consideration of a work schedule As users normally only work on projects during business times, predictions for a project's finish date must be tied to a work schedule. The work schedule consists of a list of weekdays on which the user works and the hours at which the work starts and ends. The work schedule is stored in a database table to be configurable. A more detailed schedule configuration might be required for a consumer product, but the working days and start and end hours are sufficient for the prototype. As the work schedule is used for calculations in multiple parts of the application, a `WorkScheduleUtil` was implemented that can handle each combination of two of the three properties starting date, finish date and duration and calculate the missing property respectively in compliance with the work schedule. For example, the work schedule could be configured as Monday to Friday from 9 am to 5 pm. Consequently, the calculated finish date for a task that starts on Monday at 11 am and takes 8 hours is Tuesday at 11 am.

Calculation of remaining duration For *open* tasks, the remaining duration is the duration configured by the user. If no duration is defined, the duration is assumed to be 0 minutes. For tasks *in progress* the assumed remaining duration is reduced by the time they already are *in progress*. This time is determined by finding the latest event that signifies the task being moved from the *open* stage into the *in progress* stage. The remaining duration is calculated as the difference in minutes between the event's timestamp and the current time, using the `WorkScheduleUtil` mentioned in the previous section.

Calculation of `startedAt` and `finishedAt` based on work schedule Whenever the constraint solver assigns a new planning variable, the shadow variables `startedAt` and `finishedAt` have to be calculated. Therefore, the `WorkScheduleUtil` is used to be compliant with the work schedule. As already mentioned above, for `startedAt` the latest out of the task's start date and the previous task's finish date (`finishedAt`) is used. If the resulting date does not lie within the work schedule, the next possible date within the work schedule (e.g. the next day at the start of the working hours) is chosen. `finishedAt` is calculated by adding the assumed remaining duration to `startedAt` considering only timeslots within the work schedule.

4.3 Web application: Technology stack

The client of the prototype consists of a web application that runs in any modern web browser. It is written in TypeScript [73], a typed programming language that builds on JavaScript. The application was built using the framework Next.js [83], which is based on React [64], a component-based JavaScript library for building user interfaces, Babel [68], a JavaScript compiler that enables backwards compatibility for source code and Webpack [66], a JavaScript module bundler and code splitter. Next.js allows an easy set-up and configuration of the application and enables optimisation possibilities such as prerendering to improve page load times. In order to build the design of the user interface, Tailwind CSS [67] was integrated, a utility-first CSS framework that allows styling elements directly in the HTML markup by providing predefined classes.

A user study was conducted to examine the application of the prototype in practice. The user study aimed to evaluate the concepts and usability of the prototype and identify areas of improvement for future iterations. The characteristics, study design and results of the user study are described in this chapter.

5.1 Methodology

5.1.1 Study design

The user study was conducted as a within-subjects study in a controlled laboratory environment, guided and observed by the study supervisor. Thereby, data was collected in a mixture of attitudinal and behavioural approaches. Furthermore, qualitative and quantitative feedback was collected with questionnaires. The study took 45 to 60 minutes to complete. It consisted of four different tasks split into two different parts. Most of the tasks were followed by a questionnaire with task-specific questions. Elements in the software represented the tasks. For each task, a new workspace was used (see section 3.3.1). Participants were allowed to ask questions throughout the study.

5.1.2 Procedure

Table 5.1 presents an overview of the study's procedure. The following explains the study's parts in detail.

5.1.2.1 Intro

During the introduction phase of the study, *Questionnaire 1* had to be completed, which included general questions regarding demography and prior experience with board-based tools. An exploration phase followed to give the participants an overview feature-wise and familiarise themselves with the interface. The exploration included creating, modifying, reordering, moving and converting elements. While the participants were in control

Phase	Name	Description	Time limit
Intro	Questionnaire 1	Questions concerning demography and prior experience with board based tools	-
	Exploration	Guided exploration of software	
Part 1	Task 1.1	Create custom arrangement for predefined tasks of example project	10min
	Ext. NASA-TLX	Concerning Task 1.1	
	Questionnaire 2	Concerning Task 1.1	
	Task 1.2	Select favourite arrangement out of three example arrangements and personal arrangement created in Task 1.1	5min
	Questionnaire 3	Concerning Task 1.2	
Part 2	Task 2.1	Manually schedule 5 tasks with 8h total duration, 3 due dates, 1 start date	3min
	Task 2.2	Manually schedule 10 tasks with 40h total duration, 4 due dates, 3 start dates	6min
	Questionnaire 4	Concerning Task 2.1 & Task 2.2	
Outro	Questionnaire 5	General questions about software use	

Table 5.1: Study procedure.

and encouraged to experiment with the interface, the study supervisor guided them to ensure they did not miss important functionality. Also, relevant concepts were explained if participants were not familiar with board-based tools and the Kanban method.

5.1.2.2 Part 1 - Arbitrarily nested lists

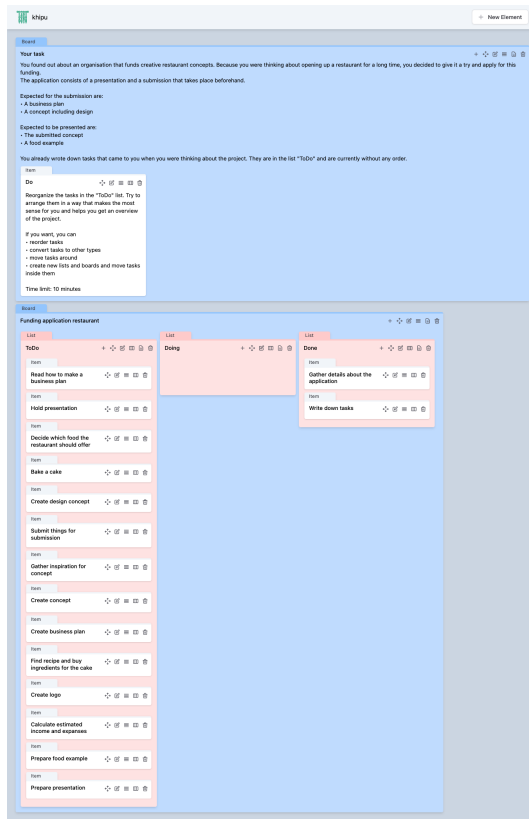


Figure 5.1: Example project of Task 1.1.

The first part of the study aimed to evaluate to which extent users use the possibility to arrange elements with unbounded nesting. Therefore, in Task 1.1 users were instructed to imagine a project consisting of the application for a funding. The application included a concept for a restaurant and a presentation. A Kanban board with a "ToDo"-list contained 14 predefined tasks presented as Items. Task 1.1 is depicted in figure 5.1. Participants had 10 minutes to arrange the project's tasks in a any way they preferred for working on that project. They were allowed to use all of the features they learned about in the exploration phase. After finishing Task 1.1 the participants filled out a extended version of the NASA-TLX questionnaire (described in more detail in section 5.4.1) followed by Questionnaire 2 containing Task 1.1 specific questions.

The following Task 1.2 presented three different boards containing three different possible arrangements for the project of Task 1.1. The arrangements are displayed in

figure 5.2. The main difference between the arrangements is the maximum nesting depth starting from the standard depth of 2 (Board A, displayed in figure 5.2a) to a maximum nesting depth of 5 (Board C, shown in figure 5.2c). Board C additionally uses not only Lists in a nested way but also Boards. The participants were instructed to look at the arrangements for a maximum of 5 minutes and then fill out *Questionnaire 3* to answer which of the representations they liked the most. Furthermore, they were asked to compare this representation to their personal solution and select an overall favourite representation.

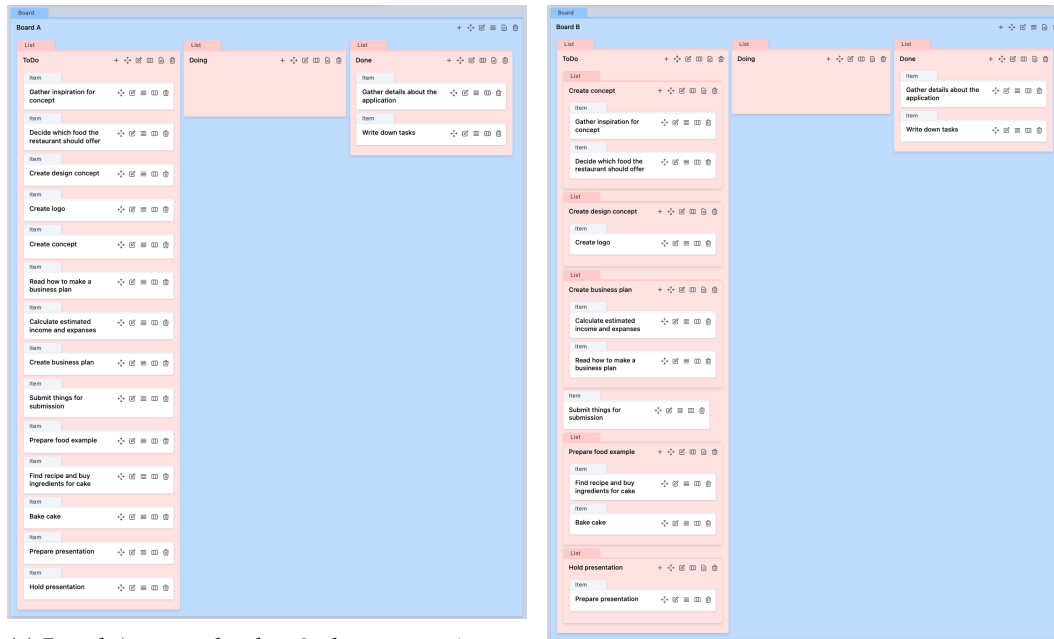
5.1.2.3 Part 2 - Symbolic intelligence

The second part of the study focused on the symbolic intelligence of the software in the form of the scheduling feature. The full description of *Task 2* can be found in figure 5.3. *Task 2.1* depicted in 5.3a presented a total of 5 Items representing tasks. While all items had temporal resources in the form of an estimated duration attached, three items additionally had either start date and due date or both configured. The total duration of the tasks was 8 hours and would therefore span the project over one day in the best-case scenario. Also, some constraints were given. The current time was assumed to be Monday, 1 November 2021, 10:00 am, and the work schedule consisted of working Monday to Friday between 10:00 am and 06:00 pm. Furthermore, only one task could be processed simultaneously, and splitting tasks was not allowed. It was reassured that participants correctly understood the temporal constraints: start date, due date, and duration.

The objective was to estimate an execution order of the tasks to finish the project as early as possible while minimising the number of missed due dates. The order was created by reordering the tasks in the "ToDo"-list. Participants also estimated the project's finish date and how many due dates they would miss with their execution order and wrote that estimation into the "Estimation" Item. The time limit for the estimation was 3 minutes. After the manual planning, the participants triggered the scheduling feature provided by the prototype, perceived the result, and compared it to their manual planning. Subsequently, *Task 2.2*, which is presented in figure 5.3b followed. While the objective and procedure of this task were equivalent to *Task 2.1*, this time, the board consisted of a total of 10 Items representing tasks, thus making the planning significantly harder. The total duration of these items was configured to 40 hours, which made them span over five days in a best-case scenario. The time limit for the estimation was 6 minutes. After *Task 2.1* and *Task 2.2* the participants filled out *Questionnaire 4* which contained questions related to these tasks.

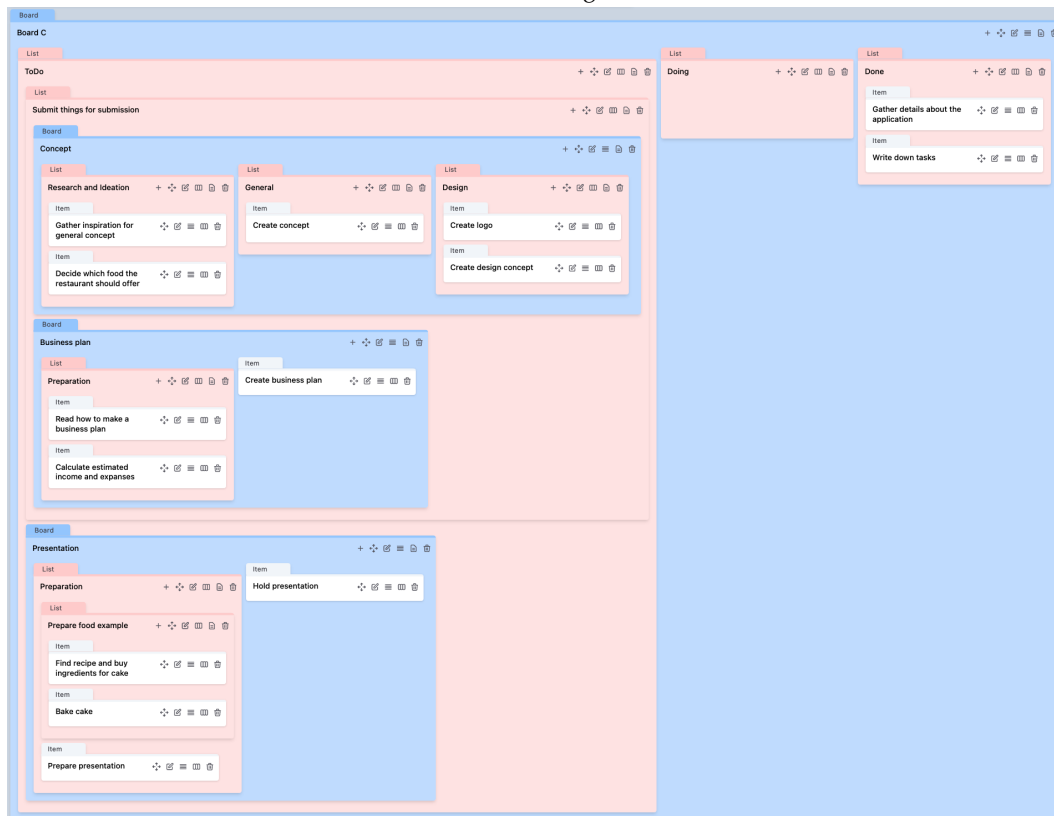
5.1.2.4 Outro

Questionnaire 5 was the final objective to complete. It included questions regarding the application's usability throughout all tasks, and participants gave general feedback. Throughout all questionnaires, whenever questions with free text answers occurred, the participants were encouraged to think a load, which allowed the study supervisor to understand better their answers and ask follow-up questions if necessary.



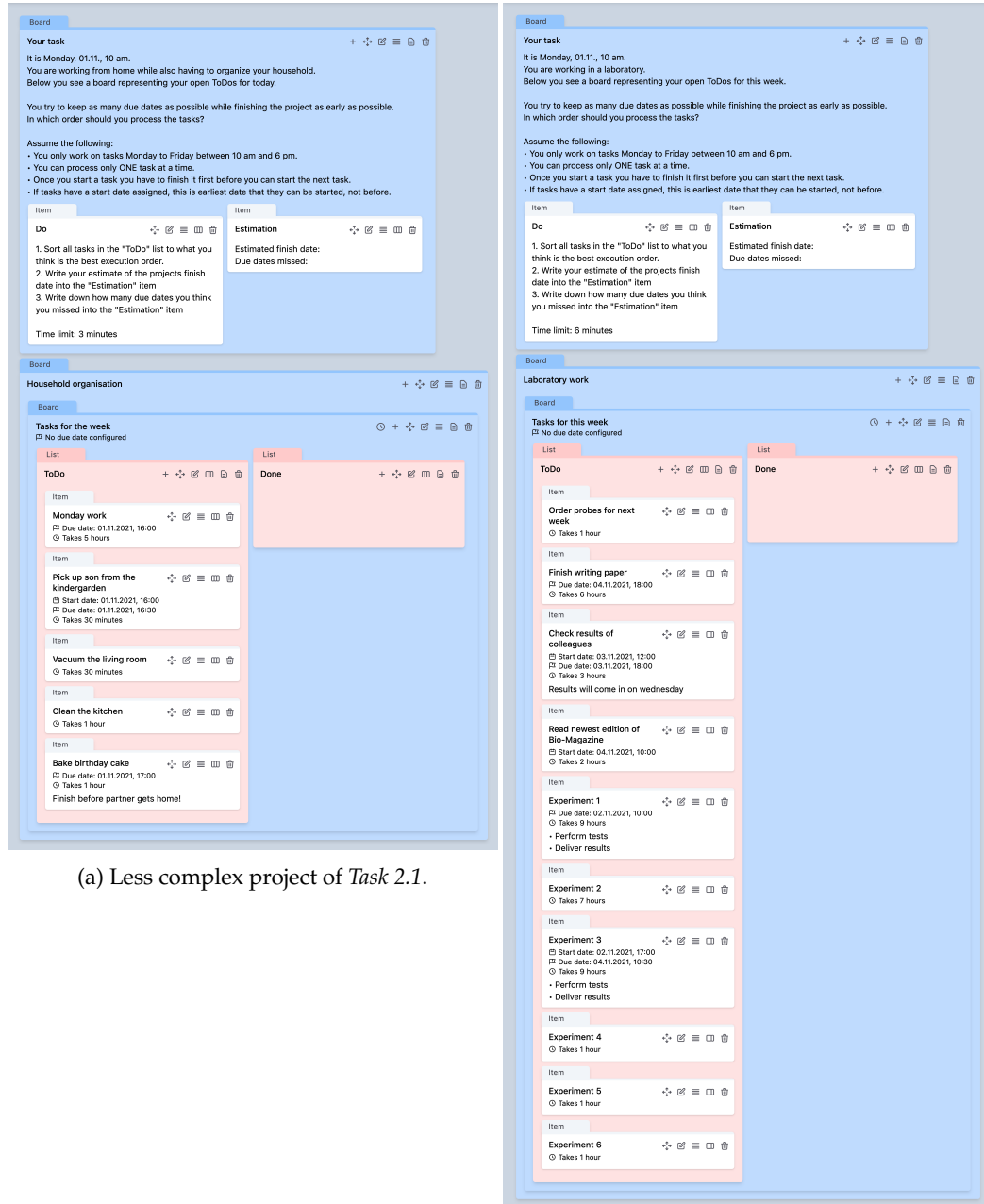
(a) Board A, max. depth = 2, deepest nesting arrangement: Board \rightarrow List \rightarrow Item.

(b) Board B, max. depth = 3, deepest nesting arrangement: Board \rightarrow List \rightarrow List \rightarrow Item.



(c) Board C, max. depth = 5, deepest nesting arrangement: Board \rightarrow List \rightarrow List \rightarrow Board \rightarrow List \rightarrow Item.

Figure 5.2: Example arrangements presented in *Task 1.2*.



(a) Less complex project of Task 2.1.

(b) More complex project of Task 2.2.

Figure 5.3: Task 2 with the objective to estimate two different sized projects.

5.2 Participants

Thirteen participants (two female) were recruited for this study. Their median age was 27, with ages ranging from 25 to 30. All participants had normal vision or corrected to normal vision. All participants had the academic background of an Abitur (German equivalent of "A Levels"), with 12 participants having completed a university degree. Ten participants (77%) had used board based tools before, while three participants (23%) did not have prior experience with board-based tools. The ten participants that had used board based tools before rated their prior experience with board based tools with 5.9 on average (on a scale from 1 to 10). When being asked which tools they had used before, they mentioned Trello ($n = 8$, [56]), Atlassian Jira ($n = 4$, [54]), Asana ($n = 1$, [62]), Microsoft Teams ($n = 1$, [72]), Miro ($n = 1$, [74]), Notion ($n = 1$, [75]) and YouTrack ($n = 1$, [65]). Five participants use board based tools during weekdays, three participants use them once a week, and two participants once a month. When asked in which areas board-based tools are used, they answered work ($n = 10$), school or university ($n = 4$), personal projects ($n = 2$), and other areas ($n = 2$).

5.3 Limitations

Sample size and demography Due to the circumstances of this work, only a small number of participants was recruited. Thereby, the expressiveness of the statistical evaluation is limited. Also, the study can not be considered representative and does not reflect the demography of Germany as all participants came from an academic background.

Study design In the study, participants spend little time with the software. As the concept of having arbitrarily nested lists is novel, it might take more time to get used to it. Moreover, the software is designed to manage long-term projects. However, the study provides little insight into long-term usage, as the tasks only concerned initial project planning phases. Finally, while participants were encouraged to imagine real-life problems, the controlled environment limits the study's expressiveness for real-life application.

5.4 Measurements

5.4.1 Quantitative measures

Task completion time During *Task 1.1*, *Task 2.1* and *Task 2.2* the time it took the participants to complete the tasks was measured manually.

Extended NASA-TLX An extended version of the NASA-TLX [22] was utilised to measure workload, sense of control and creativity support of *Task 1.1*. It firstly consists of the Task-Load Index (TLX) from NASA [22], which measures the perceived workload experienced by the participants during the task. Additionally, the sense of control scale by Dong et al. [19] measures the perceived control during the task. In order to be consistent with the scale of the NASA-TLX, its original 6 point rating scale was adapted to a 20-point rating scale as in the NASA-TLX. Lastly, the Creativity Support Index by Cherry et al. [16] measures how well an application supports the creativity of the users. It was limited to the dimensions of exploration, motivation and enjoyment.

Likert-type questions Likert-type questions that can be answered using five response alternatives (strongly disagree, disagree, neutral, agree, strongly agree) were used to gather more implicit quantitative data about how users perceived *Task 1.1*, *Task 2.1*, *Task 2.2* and the software in general [14].

Number of actions The number of actions performed by the users during *Task 1.1* was measured to gather insights into how users with different experience levels are using the application. As described in chapter 3, these actions were automatically stored in the database.

Maximum nesting depth The maximum nesting depth of the personal arrangement created by the users in *Task 1.1* was measured by counting the nesting level from the root level to the deepest nested element. In a standard Kanban board, an Item is nested into a List, which is nested into a Board, which results in a total of two nesting operations. Consequently, the maximum nesting depth for a standard Kanban board is 2.

Task scheduling estimation The following measures were collected to evaluate the manual scheduling created by participants in *Task 2.1* and *Task 2.2* :

- Estimated best execution order of the tasks created by the participants: The order of the tasks in the "ToDo" list, with the first Item in the list being the first task to be processed and so forth.
- Estimated finish date of the project when executing the tasks in the estimated order: The participants entered their answers in the "Estimation" Item.
- Estimated number of due dates missed when executing the tasks in the estimated order: The participants entered their answers in the "Estimation" Item.

5.4.2 Qualitative measures

Several open-ended questions (see table 5.2) were used across the questionnaires to gather qualitative data about how the participants used the application. Moreover, the participants were observed during the study and noticeable events, like certain behaviours, were recorded.

5.5 Data analysis and results

In the following, the results of the user study are presented. R, a programming language for statistical computing and graphics, was used for most of the data analysis as well as the creation of the figures [81].

5.5.1 Quantitative data

5.5.1.1 Part 1 - Arbitrarily nested lists

Time required On average it took the participants 07:27 minutes to complete *Task 1.1*. The maximum allowed time was 10 minutes.

Actions Figure 5.4 shows the mean of number of actions performed by the participants in order to create their personal arrangement in *Task 1.1*. The mean of all actions performed by the participants was 38 ± 13.24 . The most used action was MOVE (mean = 18.62 ± 7.38) followed by REORDER (mean = 9.08 ± 6.29) and CONVERT (mean = 4.15 ± 4.00). One participant discovered the resources functionality and used it to configure three temporal resources. The mean number of actions was also grouped by prior experience with board based tools. For participants without prior experience the mean of total actions was lower (32.00 ± 13.00 , $n = 3$) than the mean for participants with prior experience (39.50 ± 13.48 , $n = 10$). When looking at the specific actions, participants without prior experience had a lower mean in MOVE actions (14.33 ± 4.93 , $n = 3$ vs. 19.90 ± 7.69 , $n = 10$), CREATE actions (0.67 ± 1.15 , $n = 3$ vs. 5.00 ± 3.68 , $n = 10$). On the other, no prior experience resulted in a higher mean for REORDER actions (10.33 ± 4.51 , $n = 3$ vs. 8.70 ± 6.90 , $n = 10$) and CONVERT actions (5.33 ± 6.11 , $n = 3$ vs. 3.80 ± 3.61 , $n = 10$).

The unpaired two-samples t-test and the unpaired two-samples Wilcoxon test were used depending on the normal distribution of the data to validate the statistical significance of the differences between prior experience and no prior experience. However, only the difference in medians of the CREATE action showed to be significant with $p < .05$. This might be related to the small sample size ($n = 3$) of participants without prior experience.

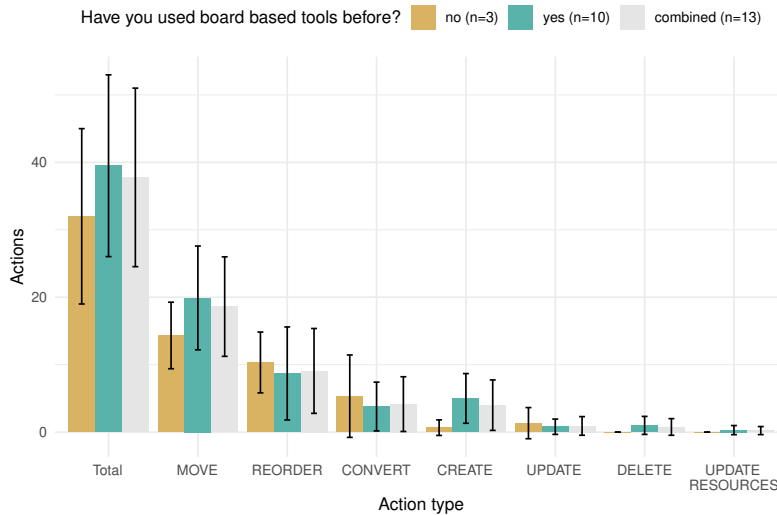


Figure 5.4: Average number of actions by action type by prior experience.

Extended NASA-TLX Figure 5.5 depicts the results of the extended NASA-TLX questionnaire. In order to unify the scales of the extended NASA-TLX regarding workload, sense of control, and creativity support, the scale of the workload was adjusted to 0-100 from 0-600 and the scale of creativity support to 0-100 from 0-300. Figure 5.5 shows the results of the extended NASA-TLX questionnaire about *Task 1.1*. On average, the participants rated the workload intensity during the task at 40.64 ± 9.30 . Furthermore, the participants rated the sense of control during the task at 75.77 ± 21.78 . Finally, they classified their subjective perception of creativity support at 73.72 ± 15.13 .

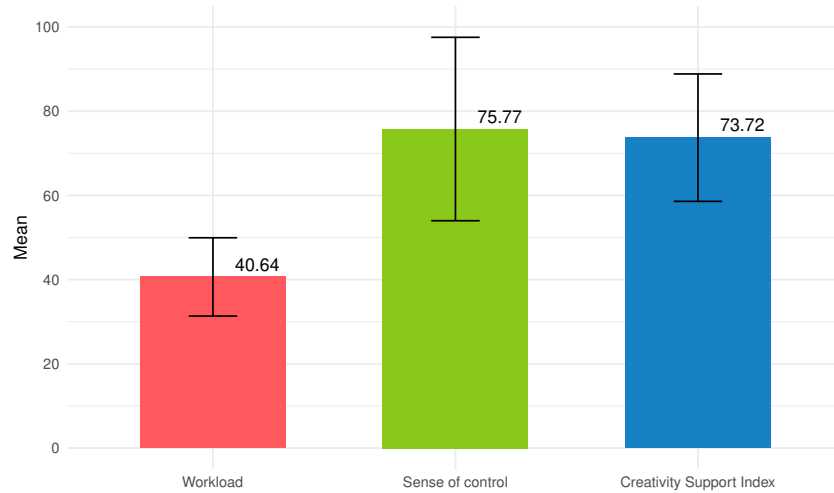


Figure 5.5: Results of Extended NASA-TLX ($n = 13$).

Likert-type questions Figure 5.6 depicts the result of the three Likert-type questions about *Task 1.1*. 100% of the participants agreed or strongly agreed that they understood the task's objective. Also, 100% felt supported by the software in the process of visualising the project. Finally, 85% agreed or strongly agreed that the result they created matched their mental image of the project, while 15% neither agreed nor disagreed.

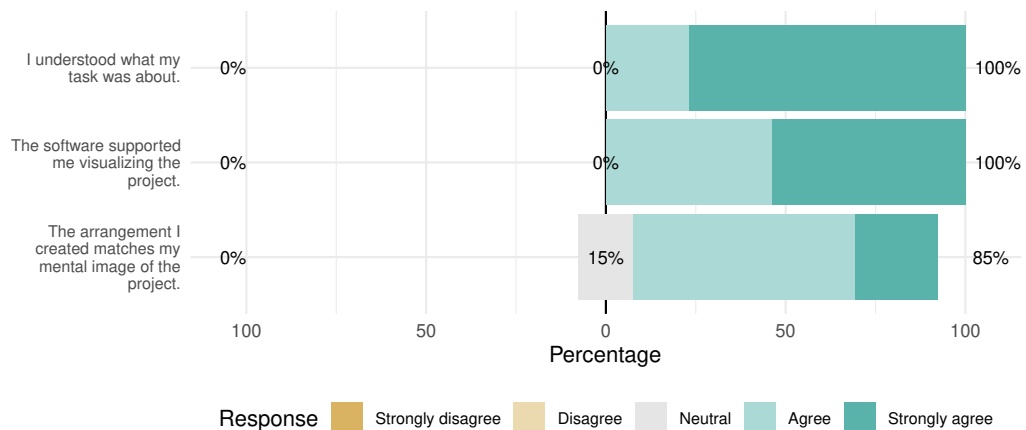


Figure 5.6: Likert-type questions about *Task 1.1* ($n = 13$).

Maximum nesting depth Figure 5.7a depicts the maximum nesting depths of the personal arrangements created in *Task 1.1*. It can be seen that participants created one arrangement with a maximum depth of 5, three arrangements with a maximum depth of 4, five arrangements with a maximum depth of 3 and four arrangements with a maximum depth of 2. Figure 5.7b represents which arrangement the participants favoured among the three examples presented in *Task 1.2*. Six people (46.2%) favoured Board C, which had a maximum nesting depth of 5. Five participants (38.5%) preferred Board B with a maximum depth of 3. Two people (15.4%) preferred Board A with a standard nesting depth of 2. When looking at the overall favourite board of the participants as shown in figure 5.7c, two participants (15.4%) preferred their personal solution created in *Task 1.1* over the example arrangements.



(a) Personal arrangements created in *Task 1.1*.



(b) Favoured board among examples of *Task 1.2*.



(c) Favoured board among examples of *Task 1.2* and arrangements created in *Task 1.1*.

Figure 5.7: Occurrences of maximum nesting depths among participant's personal and favoured arrangements ($n = 13$).

Figure 5.8 displays a comparison of the standard nesting depth of 2 to the maximum nesting depth of the personal arrangements (mean= 3.08 ± 0.95 , median=3), the maximum depth of the favoured example arrangement (mean= 3.77 ± 1.24 , median=3) and the maximum depth of the overall favoured arrangement including examples and personal arrangement (mean= 3.77 ± 1.30 , median=4).

The personal nesting depths were normally distributed, as assessed by Shapiro-Wilk's test ($p > .05$) and a Normal Q-Q plot. Also, there were no outliers in the data, as assessed by inspection of a boxplot. The one-sample t-test shows that the mean of the personal nesting depths was significantly greater than the standard nesting depth of 2 ($t(12) = 4.0698$, $p = .0007771$).

For the favoured example arrangement and the favourite overall, the nesting depths were not normally distributed, as assessed by Shapiro-Wilk's test ($p < .05$). Therefore, a one-sample Wilcoxon signed-rank test was performed for each case. It shows that the medians of 3 resp. 4 were significantly greater than the standard nesting depth of 2 with $p < .05$.

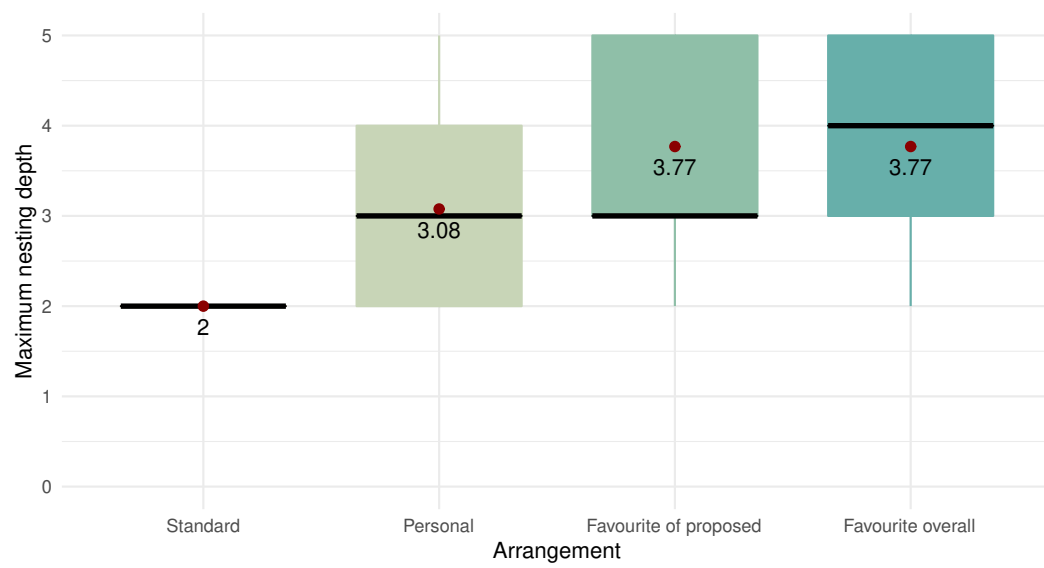


Figure 5.8: Maximum nesting depth by arrangement ($n = 13$).

5.5.1.2 Part 2 - Symbolic Intelligence

Time required The average time it took the participants to finish their estimation was 02:22 minutes for *Task 2.1* (maximum time: 3:00 minutes) and 05:41 minutes for task *Task 2.2* (maximum time: 6:00 minutes).

Scheduling performance Two error types were defined to evaluate the participant's manual scheduling performance.

- The **assessment error** is defined as the delta between the estimation of the participants and the evaluation of their manually created task execution order. This evaluation, calculated by the backend, consists of the finish date and the number of missed due dates of that order.

The assessment error for the finish date is defined as

$$E_{\text{assessment}}(F_e, F_r) = |\Delta_w(F_e, F_r)| \quad (5.1)$$

where F_e = estimated finish date, F_r = resulting finish date, Δ_w = difference in working hours.

An assessment error of 0 implies that a participant correctly estimated the resulting finish date of their manual order. An assessment error of 4 signifies that a participant's estimation of the finish date differed by four working hours from the finish date of their order. Δ_w describes the difference **within the given work schedule**. For the given work schedule of working Monday to Friday from 10am to 6pm the assessment error is calculated exemplary: $E_{\text{assessment}} = (2021-11-01-18:00, 2021-11-02-14:00) = 4h$

The assessment error for the missed due dates is defined as

$$E_{\text{assessment}}(D_e, D_r) = |\Delta(D_e, D_r)| \quad (5.2)$$

where D_e = estimated missed due dates, D_r = resulting missed due dates.

An assessment error of 0 means that the participant correctly estimated the number of missed due dates that resulted from their manual order. An assessment error of 1 means that the participant's estimation of missed due dates differed by one due date from the number of missed due dates that their order caused.

- For the **true error** the evaluation of the participants manual order was compared with the evaluation of an optimal order.

The true error for the finish date is defined as

$$E_{\text{true}}(F_r, F_o) = |\Delta_w(F_e, F_r)| \quad (5.3)$$

where F_r = resulting finish date, F_o = optimal finish date, Δ_w = difference in working hours.

A true error of 0 implies that the participants order led to the earliest possible finish date.

The true error for the missed due dates is defined as

$$E_{\text{true}}(D_r, D_o) = |\Delta(D_e, D_r)| \quad (5.4)$$

where D_e = resulting missed due dates, D_r = optimal missed due dates.

A true error of 0 signifies that the participants order led to the minimum possible number of missed due dates.

Finish date The error of the finish date in working hours can be found in figure 5.9. Figure 5.10 displays the occurrences for the estimated and resulting finish dates as well as the optimal finish date. Each line in a non-horizontal direction indicates an assessment error > 0 for the participant.

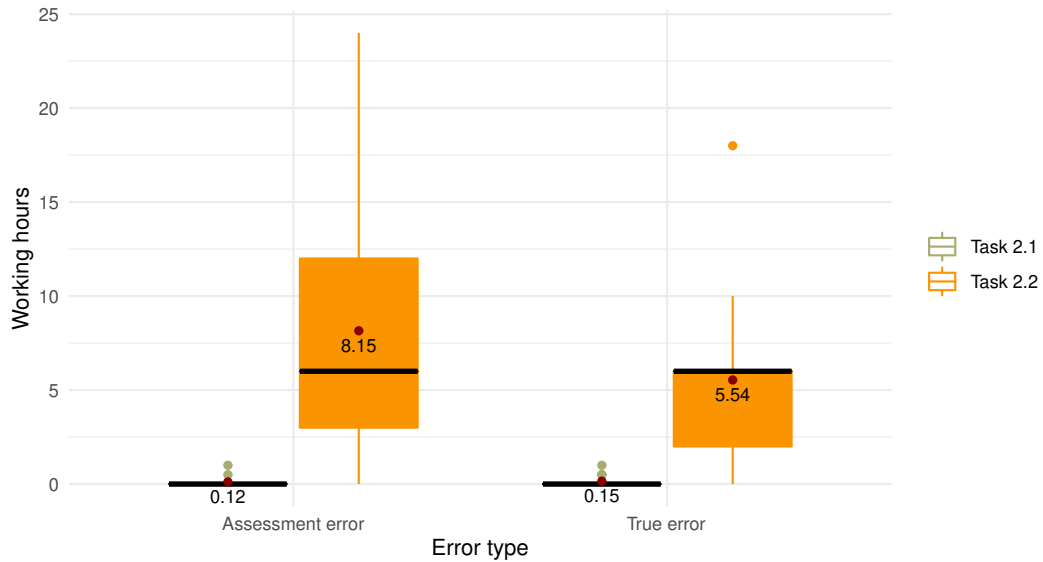


Figure 5.9: Error in working hours by task by error type ($n = 13$).

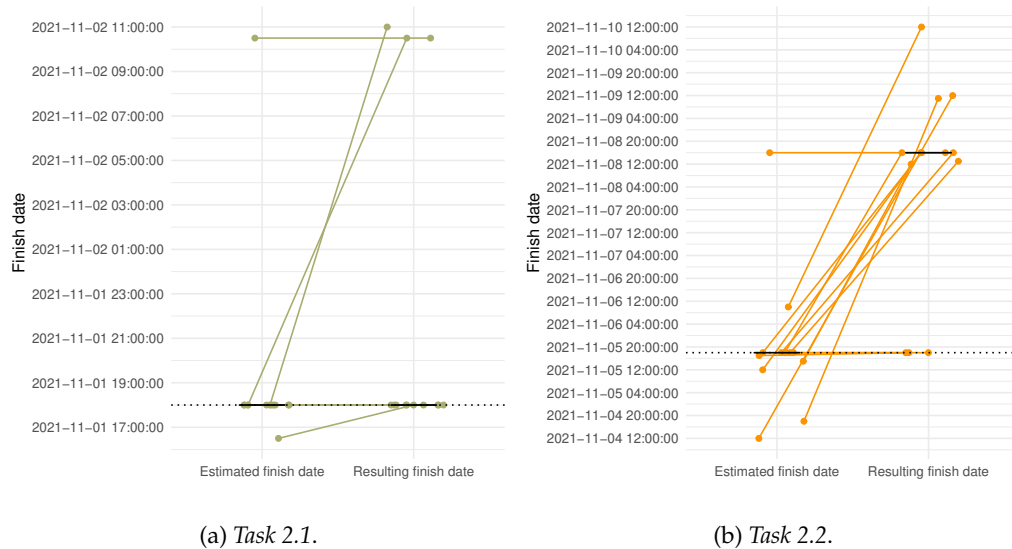


Figure 5.10: Paired scatter plot showing the occurrences of the estimated and resulting finish date ($n = 13$). The dotted line indicates the optimal finish date. The black line indicates the median.

Assessment error In the less complex *Task 2.1* **eleven** participants estimated the finish date of their manual planning order correctly. The number led to a mean assessment error

of 0.12 ± 0.35 working hours. In the more demanding *Task 2.2* **two** participants estimated the finish date of their manual planning order correctly. This number resulted in a higher mean assessment error of 8.15 ± 7.50 working hours.

Statistical tests were performed to validate the statistical significance of the assessment error. Dates were converted to numeric values for the tests. The absolute difference (not in working hours) between the estimated and resulting finish dates of *Task 2.1* and *Task 2.2* was not normally distributed, as assessed by Shapiro-Wilk's test ($p < .05$). Therefore, a paired samples Wilcoxon test was performed. It shows, that in *Task 2.1* the median of the estimated finish dates (2021-11-01-18:00:00) didn't deviate significantly from the median of the resulting finish dates (2021-11-01-18:00:00) ($p > .05$). However, in *Task 2.1* the median of the estimated finish dates (2021-11-05-18:00:00) was significantly smaller than the median of the resulting finish dates (2021-11-08-16:00:00) ($p < .05$).

True error In the less complex *Task 2.1* **ten** participants created an order that led to the optimal finish date, which was 2021-11-01 18:00:00. This number led to a mean true error of 0.15 ± 0.32 working hours. In the more demanding *Task 2.2* **three** participants created an order that led to the optimal finish date, which was 2021-11-05 18:00:00. This number resulted in a higher mean true error of 5.54 ± 4.99 working hours.

Statistical tests were performed to validate the statistical significance of the true error. Dates were converted to numeric values for the tests. The resulting finish dates of *Task 2.1* and *Task 2.2* were not normally distributed, as assessed by Shapiro-Wilk's test ($p < .05$). Therefore, a one-sample Wilcoxon signed-rank test was performed. It shows that in *Task 2.1* the median 2021-11-01-18:00:00 did not deviate significantly from the optimal finish date 2021-11-01-18:00:00 ($p > .05$). However, in *Task 2.1* the median 2021-11-08-16:00:00 was significantly greater than the optimal finish date 2021-11-05-18:00:00 ($p < .05$).

Missed due dates The error in missed due dates can be found in figure 5.11. The occurrences for estimated and the resulting number of missed dates as well as the optimal number of missed due dates are depicted in figure 5.12. Each line in a non-horizontal direction indicates an assessment error > 0 for the participant.

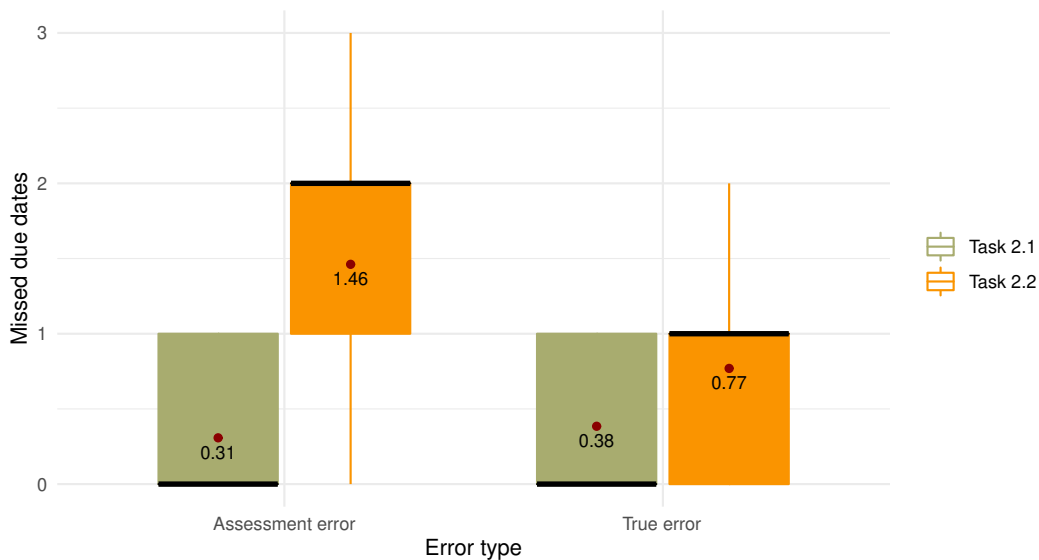


Figure 5.11: Error in missed due dates by task by error type ($n = 13$).

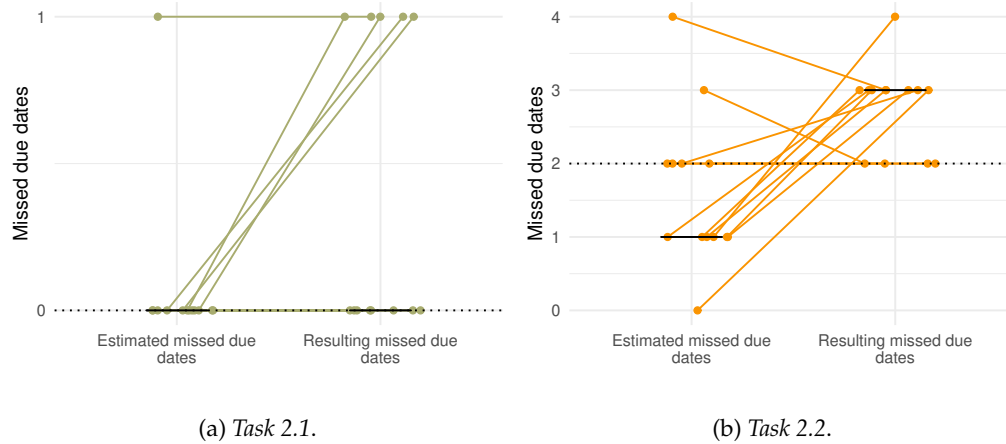


Figure 5.12: Paired scatter plot showing the occurrences of the estimated and resulting missed due dates ($n = 13$). The dotted line indicates the optimal number of missed due dates. The black line indicates the median.

Assessment error In the less demanding *Task 2.1* **nine** participants estimated the number of missed due dates of their manual planning order correctly. This number resulted in a mean assessment error of 0.31 ± 0.48 . In the more complex *Task 2.2* **three** participants estimated the number of missed due dates of their manual planning order correctly. This number resulted in a higher mean assessment error of 1.46 ± 1.05 missed due dates. As for the finish date, statistical tests were performed to validate the statistical significance of the assessment error. The absolute difference between the estimated and the resulting missed due dates in *Task 2.1* was not normally distributed, as assessed by Shapiro-Wilk's test ($p < .05$). Therefore, a paired samples Wilcoxon test was performed. It shows that in *Task 2.1* the median of the estimated missed due dates (0) did not deviate significantly ($p > .05$) from the median of the resulting missed due dates (0). In *Task 2.1* the difference between the estimated and the resulting missed due dates was normally distributed as assessed by Shapiro-Wilk's test ($p > .05$) and a Normal Q-Q plot. Also, there were no outliers in the data as assessed by the inspection of a boxplot. Therefore paired samples t-test was performed. It shows that in *Task 1.2* the mean of the estimated missed due dates (1.62) was significantly smaller than the mean of the resulting missed due dates (2.77) ($t(12) = -2.9608, p = .005952$).

True error In the less demanding *Task 2.1* **eight** participants created an order that led to the optimal number of 0 missed due dates. This number resulted in a mean true error of 0.38 ± 0.51 missed due dates. In the more complex *Task 2.2* **four** participants created an order that led to the optimal number of 2 missed due dates. This number resulted in a higher mean true error of 0.77 ± 0.60 missed due dates. Statistical tests were performed to validate the statistical significance of the true error. The resulting missed due dates of *Task 2.1* and *Task 2.2* were not normally distributed, as assessed by Shapiro-Wilk's test ($p < .05$). Therefore, a one-sample Wilcoxon signed-rank test was performed. It shows, that in *Task 2.1* the median of 0 didn't deviate significantly ($p > .05$) from the optimal 0 due dates missed. However, in *Task 2.1* the median 3 was significantly greater ($p < .05$) than the optimal 2 due dates missed.

Order of tasks Figure 5.13 displays the similarity of the task execution order that was created by the participants to the proposed and the closest optimal order. As a similarity

measure, the Gestalt Pattern Matching Algorithm developed 1983 by Ratcliff and Oberhelp [41] was used, which returns a similarity value $0 \leq S \leq 1$. 0 signifies 0% similarity, and 1 implies 100% similarity. The Python implementation `difflib.SequenceMatchers` [58] was utilised to calculate the similarity.

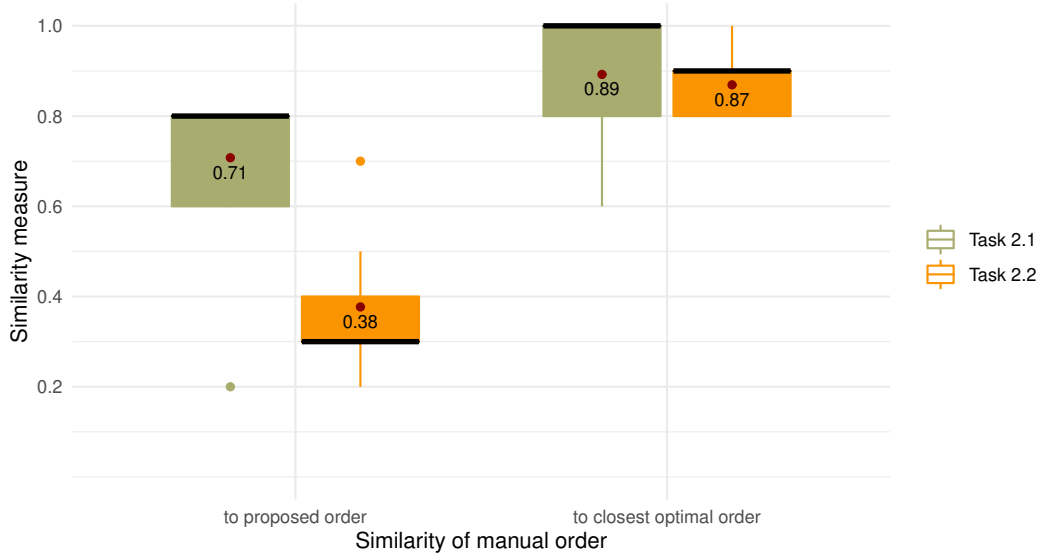
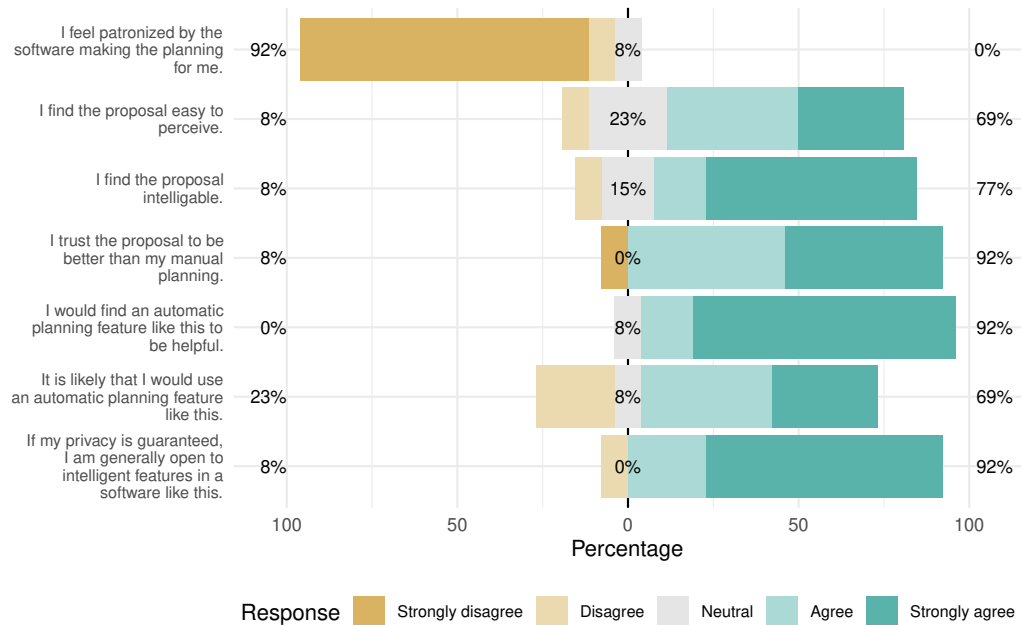


Figure 5.13: Comparison of similarity of estimated execution order grouped by task ($n = 13$).

In the less complex *Task 2.1* **eight** participants created an optimal order. The mean similarity of the order that the participants created to the order they got proposed was 0.71 ± 0.18 . The mean similarity to the closest optimal order was 0.89 ± 0.16 . The lower similarity to the proposed order can be explained by the fact that there were four possible optimal orders from which one the algorithm randomly proposes, so the probability for getting an optimal matching order was only $\frac{1}{4}$.

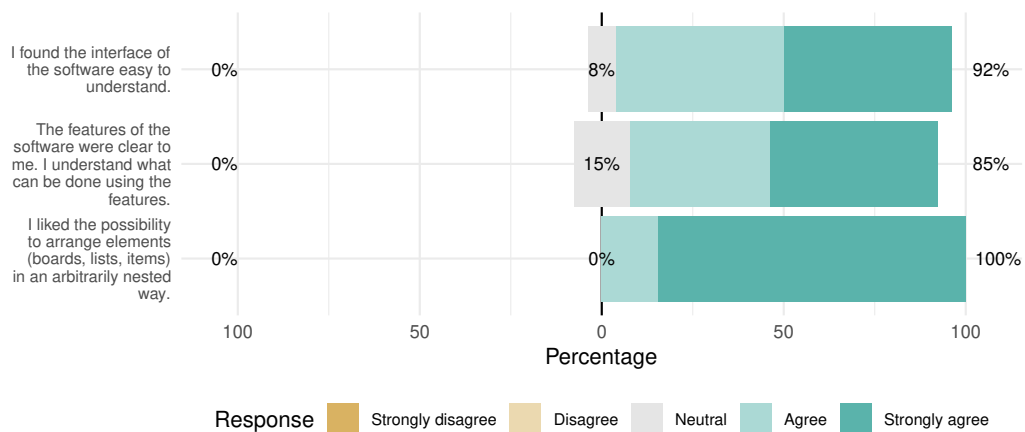
In the more demanding *Task 2.2* **one** participant created an optimal order. The mean similarity of the order that the participants created to the order they got proposed was 0.38 ± 0.18 . The mean similarity to the closest optimal order was 0.87 ± 0.06 . In total, there were 48960 optimal orders. Therefore, the probability of getting a proposed order that matches the created order was very low. However, with that many optimal orders, it was very likely to create an order that matches one of these optimal orders by 80% or 90%, which results in a high similarity value to the closest optimal order.

Acceptance of intelligence Figure 5.14 depicts the result of the seven Likert-type questions about *Task 2.1* and *Task 2.2*. 92% of the participants disagreed or strongly disagreed that they felt patronised by the scheduling system. 69% agreed or strongly agreed that they found the proposed order easy to perceive. 77% found the proposed order intelligible. 92% trusted the proposal to be better than their manual order. Also, 92% find an automatic planning feature like this helpful. 69% would use a planning feature like this. Finally, 92% agreed or strongly agreed to be generally open to intelligent features in software like this if their privacy was guaranteed.

Figure 5.14: Likert-type questions about *Task 2* ($n = 13$).

5.5.1.3 General usability

Figure 5.15 shows the result of the three Likert-type questions in the final questionnaire, which contained questions about the software as a whole. 92% of the participants agreed or strongly agreed that they found the software interface easy to understand. 85% understood the features of the software. Finally, 100% liked the possibility to arrange elements in an arbitrarily nested way.

Figure 5.15: Likert-type questions after finishing all tasks ($n = 13$).

5.5.2 Qualitative data

5.5.2.1 Questions

Table 5.2 gives an overview of the qualitative questions asked throughout the study.

Qualitative questions		
Q01	Part 1	Which feature(s) helped the most in achieving your task?
Q02	Part 1	Why did you select Board X as your favourite?
Q03	Part 1	Why did or did you not favour your personal arrangement over the examples?
Q04	Part 2	Do you have any specific intelligent feature for this application in mind that you would like to have?
Q05	Part 2	Do you have any general comments regarding the planning feature?
Q06	General	What did you like when using the software?
Q07	General	What didn't you like when using the software?
Q08	General	Would you use a software like this in the future? If yes, what would you use it for and in which part of your life? If no, why not?
Q09	General	Are there any features you were missing or things I should add or improve?
Q10	General	Do you have any general feedback?

Table 5.2: Overview of the qualitative questions throughout the questionnaires.

Part 1 For question **Q01** participants stated that the features that helped them the most in creating their personal arrangement in *Task 1.1* were arbitrary nesting ($n = 4$), the different types of elements, Board, List and Item ($n = 3$), converting elements ($n = 2$), moving elements ($n = 2$), reordering elements ($n = 1$) and having a Kanban board ($n = 1$). For question **Q02** participants, who selected Board A, explained it as being the simplest arrangement ($n = 2$). Also, they stated that they did not know how to handle nested lists in a Kanban board, as moving lists of tasks from *Todo* into *Doing* would not make sense to them ($n = 1$). Participants who favoured Board B justified it with being a good balance between simplicity and complexity ($n = 2$), having advantages over traditional Kanban board nesting while being clear enough ($n = 2$) and being similar to their personal solution ($n = 2$). Finally, participants who have chosen Board C described it as the best presentation of the project ($n = 4$) and saw it as clear and easy to understand ($n = 2$). In question **Q03** participants, who favoured their personal arrangement over the presented arrangements explained, that their personal solution made more sense to them.

Part 2 Questions **Q04** and **Q05** concerned the symbolic intelligence of part 2. Participants stated that they would like to have the possibility of adding more constraints like priorities or dependencies between tasks ($n = 8$). Furthermore, participants stated they would like to have an option to improve visualisation of the temporal constraints, which could consist of a timeline or calendar view ($n = 4$). Also, they would like a better presentation of the proposed order ($n = 3$), for example, prominently highlighting the next task, a better visual comparison between the current order and proposed order or the option to accept the proposed order automatically. $n = 2$ participants stated that they would like to have push notifications on start dates or due dates. Other proposals mentioned $n = 1$ times were

automatic cleanup for old tasks, automatic splitting of tasks into sub-tasks, integration into calendar apps and location-aware tasks ($n = 1$).

General For question Q06, $n = 6$ participants stated that they liked the scheduling feature. Furthermore, concerning the usability, participants answered that they enjoyed the interface ($n = 5$), they stated that the features were clear and easy to understand ($n = 4$), they liked the possibility to drag and drop elements ($n = 3$) and emphasised the good usability ($n = 2$). Regarding the task visualisation aspect of the software, they highlighted the (arbitrary) nesting ($n = 3$) and stated that the software is very customisable and allows building your own system ($n = 3$). Besides, they liked the visualisation capabilities ($n = 2$) and the Kanban method ($n = 1$).

For Q07, $n = 8$ participants stated that they did not like the restriction of the drag and drop functionality, requiring the use of the move button in certain situations. Furthermore, they did not like modifying the title or description of an element by clicking the modify button instead of directly clicking the element ($n = 4$). From a visual point of view, participants criticised that nested elements are hard to distinguish because of the similar colours, which makes the interface cluttered ($n = 4$). In addition, for $n = 2$ participants, the items were too small, causing a hard to read interface. Besides, criticism with $n = 1$ mentions regarded that converting boards or lists to items can cause data loss, splitting tasks was not possible in *Task 2.1* and *Task 2.2* and a calendar might be the more useful tool for scheduling tasks. Finally, $n = 1$ participants commented that sometimes it is beneficial if software forces a way of organising instead of giving total freedom to the user.

As for Q08, all participants stated that they would generally use the software ($n = 13$). Areas of application consisted of work ($n = 5$), household organisation ($n = 2$), private projects ($n = 2$), the scheduling as addition to a calendar ($n = 2$), collaborative working ($n = 1$), studying ($n = 1$), working at a laboratory ($n = 1$) and research ($n = 1$).

As for Q09 participants answered that they would like to see an improved design of the software ($n = 3$), more resources to items like attachments ($n = 2$), allow checkboxes in descriptions ($n = 2$), the possibility to hide child elements of boards and show them via zoom or collapsing ($n = 2$), configurable colours of elements ($n = 1$), collaborative boards ($n = 1$) and a progress bar showing the progress within a Kanban board. An additional interesting idea consisted of abstracting the Kanban method from the Board/List structure into a different data structure such as labels and making it a separate view that can be turned on and off ($n = 1$).

Answers for Q10 did not provide any important insights not mentioned in the previous answers.

5.5.2.2 Observation

While observing the participants during the study, it was noted that almost all participants initially tried to move elements to different levels via drag and drop, although the separate move button was explained in the exploration phase. When noticing drag and drop is not possible in some situations, many participants fell started using the move button for every move situation, even if drag and drop would have been possible for some of these. Also, multiple participants tried to modify an element by clicking the description or title before realising they had to use the modify button. Another interesting observation during manual scheduling in *Task 2.1* and *Task 2.1* was that participants had different approaches to coping with the situation when they noticed they could not reach a task's due date. While most participants still tried to plan the tasks to finish as closely as possible to the due date, some moved the tasks to the end rather than trying to accomplish other tasks instead.

Discussion and future work

This chapter discusses the results of the user study and their implications. Furthermore, areas of improvement for the concept and prototype implementation are suggested. Finally, three possible use cases are explained.

6.1 Discussion of the user study results

In the following, the results presented in the previous chapter are discussed.

6.1.1 Unbounded nesting

The first part of the study focused on unbounded nesting of elements in a Kanban board. The goal was to examine to which extent users would use the feature and how it impacts the software's capabilities to visualise a project.

One of the collected metrics regards the maximum nesting depth of participants' arrangements created during the study. A mean maximum depth of 3.08 was observed, significantly differing from the standard nesting depth of 2. However, arguably the maximum nesting depth is influenced by several factors.

Firstly, some factors might have positively impacted the participants to use deeper nesting. Consisting of a Kanban board having Items in the first column, the task already started with a nesting depth of 2. Therefore, reduced nesting depths were not expected, and outcoming depths would potentially be ≥ 2 . The initial state of the board and the objective of the task, which specifically mentioned the possibility of using actions such as moving and converting elements, might have influenced the participants to use the nesting functionality more thoroughly. Also, the participants were testing a prototype with the unbounded nesting as a novelty, which they were told about in the exploration phase, which might have encouraged them to use nesting. Finally, another factor concerns the example project itself, which contained hierarchical features. It consisted of general tasks like "Create business plan" and detailed tasks like "Read how to create a business plan", logically a sub-task. On the other hand, factors that might negatively have impacted the participants in using deeper nesting was the novelty of the approach, and the short time they spent using the application. Nevertheless, the difference to the standard nesting depth consisted of at least

one full level for the project. More complex projects with more tasks and hierarchical structure might benefit from unbounded nesting even more.

As an additional metric, example arrangements were presented, and participants selected their favourite arrangement. The overall favoured arrangements (including the example and personal arrangements) had a mean maximum nesting depth of 3.77. The difference to the standard nesting depth is significant and consists of close to two levels. Board C (shown in figure 5.2c) with a maximum nesting depth of 5 was favoured the most ($n = 6$). It has to be pointed out that it was the only arrangement featuring a board within a board, hence supplying visual separation through colours. Notably, only two participants favoured their arrangement over the example arrangements. Overall, the mean maximum nesting depth of the favoured arrangement was higher than the mean maximum nesting depth of the personal arrangements. This result supports the assumption that the maximum nesting depth was negatively impacted in *Task 1.1* through factors like lack of experience with the feature.

Finally, feedback was gathered through Likert-type questions regarding the visualisation process and visualisation. However, these traits were not examined using multiple Likert-type questions composed into a Likert scale as recommended in related work [14]. Therefore no clear statements can be made, and a bias for answering the single Likert-type questions can not be ruled out. The results still indicate a trend. All participants agreed or strongly agreed that the software supported them in visualising the project; 11 out of 13 agreed that the resulting arrangement matches their mental image of the project. The unbounded nesting was also perceived as the most helpful feature for achieving the first task, as named via qualitative feedback in section 5.5.2. Besides that, the software's most liked aspects included the visualisation and the flexibility in building a custom system. On the other hand, the software's disliked aspects included concerns regarding cluttered interface for deeper nesting scenarios, as elements were perceived hard to distinguish. This issue is addressed in section 6.2 where areas of improvement are discussed.

Overall, unbounded nesting was perceived positively, and the results indicate a positive impact of unbounded nesting on the software's visualisation capabilities. However, the study only examined an initial planning phase of a project. No conclusion can be drawn regarding longer periods of time when working with the unbounded nesting and applying it to the Kanban pull-system. For example, as mentioned by a participant, it is unclear how sub-elements can be moved to *in progress* without moving their parent elements to the same column. A possible solution is outlined in section 6.2.

6.1.2 Symbolic intelligence

The second part of the study focused on the recommendation feature represented through the symbolic intelligence module. The goal was to examine how well users perform in estimating different sized projects' temporal constraints and optimal task orders based on these constraints. Furthermore, the users' attitudes towards the recommendation feature in the prototype were examined.

The users' estimation capabilities of tasks that are defined by three features, start date, due date and duration, were evaluated for a less complex (five tasks with eight hours total duration, three due dates, one start date) and a more complex project (ten tasks with 40 hours total duration, four due dates, three start dates). The objective was to propose a task order such that the earliest finish date and the minimal number of due dates missed is reached. Additionally, the participants estimated when the project finishes and how many due dates are missed by using the task execution order. The participants' estimations were

close to correct for the less complex project, both for the finish date and the number of due dates missed. Also, most of the participants proposed an optimal order of tasks. For the more complex project, however, most participants did not correctly estimate the finish date and the number of due dates missed for their order. On average, the finish date was underestimated by a full workday of 8 hours. Likewise, the participants underestimated the number of missed due dates by 1.5 on average. The orders created by the participants were also relatively far from optimal. On average, their orders resulted in finish dates 5 hours later than the optimal finish date. Similarly, 0.77 due dates more were missed than with the optimal order.

Several factors might have impacted the quality of the users' estimations. Firstly, the objective of estimating a project under the given circumstances was artificial. Although the concepts of the start date, due date, and the task's constraints were explained to the participants beforehand, a lack of understanding might have negatively impacted the estimation. Moreover, the logical order of the tasks was not taken into account to determine the best order. Although the participants were told to propose the best order to reach the earliest finish date and minimise the number of due dates, it can not be ruled out that a potential logical order of tasks impacted their decisions. In summary, participants estimated the less complex project accurately, while the estimation of the more complex project differed substantially. The estimation of the second project came directly after the first, so the impact of understanding issues should have been smaller there. While the more complex project consisted of 10 tasks, real-life projects might even be significantly more complex. Therefore, accurately estimating these projects is potentially even harder for users.

Overall, the results support the recommendation feature's initial motivation; it is hard for humans to estimate temporal constraints in bigger projects. It also stresses the potential of the symbolic intelligence in supporting project scheduling, especially if more than just temporal constraints are considered, which was the most demanded feature named by participants as qualitative feedback (see section 5.5.2).

The users' attitude towards the symbolic intelligence was evaluated through Likert-type questions. The Likert-type questions are affected by the same limitation as in section 6.1.1. Nevertheless, the results give indications, which are supported by the participants' qualitative feedback. 12 out of 13 participants agreed to trust the proposal to be better than their planning and found it helpful. Also, 12 participants did not feel patronised by the intelligence. These results indicate acceptance of the symbolic intelligence and a supporting character of the feature in terms of project planning. However, only 69% agreed that they would use the feature. Qualitative feedback and observation indicate the participant's concern that not all projects are defined through temporal constraints, thus limiting the use case. Also, the expressiveness of the intelligence is limited if the elements and its temporal resources are not maintained correctly, which presents a substantial effort and might not be the case for smaller projects. Although participants generally agreed to find the recommendations intelligible and easy to perceive, slightly lower numbers of 77% and 69% indicate that the visual presentation of the recommendations can still be improved. Potential improvements for the symbolic intelligence, partly based on the participant's feedback, are discussed in section 6.2.2.

6.1.3 General usability

Evaluating the usability and user experience of the prototype was another goal of the study. Besides observation, usability was measured through various metrics. These will be discussed in the following.

Number of actions Generally, while creating the visualisation for the project in the first task, participants without prior experience with board-based tools performed fewer actions than participants with prior experience. This observation is not a surprise; arguably, it takes some time to get used to board-based tools. However, the difference was not significant, possibly partly because of a small dataset of only three participants without prior experience. Nevertheless, the metric could be an indicator regarding the user's experience with board-based tools.

Extended NASA-TLX The measures of the extended NASA-TLX are of a subjective matter. Without creating comparability by performing multiple measures throughout the study, their expressiveness is limited. However, generally high control and creativity support scale values are desired. The workload within the first task was recorded at the medium of the scale (mean 40.64), indicating a not too complicated task. Means of sense of control and creativity support were observed on the upper third of the scale (75.77 and 73.72).

Feedback and observation Most insights regarding usability are given through quantitative and qualitative feedback and observation. The Likert-type questions are affected by the same limitation as described in 6.1.1. Nevertheless, the results indicate an easy to perceive interface and easy to understand features. These indications are supported through qualitative feedback. Here, the software's liked aspects in terms of usability include the interface and easy to understand features. Besides that, the drag and drop functionality was highlighted positively. On the other hand, the drag and drop functionality's restriction was also disliked when using the software. As described in section 3.3.2, when moving elements to different levels, the move button had to be used, leading to two different actions required for the same outcome. Results of the study show that this was perceived as unintuitive and caused frustration when trying to find the correct action to use. Furthermore, using the move button required two additional clicks, thus slowing down the process. A second usability problem revealed by qualitative feedback and observation concerns the modification of elements, for which the modify button had to be clicked. Users intuitively wanted to edit the element's details in place. These two flaws should be addressed in future iterations of the prototype as described in section 6.2.

Overall the usability of the prototype was perceived positively, and areas of improvement were identified. It has to be stated that the study's expressiveness for long-term usability is limited. However, the study's goal was to evaluate the initial concept of the prototype and evaluate the usability in a more general manner. In the sense of an iterative design process, more studies examining usability more thoroughly should be conducted towards a consumer product.

6.1.4 Outlook

As mentioned above, the study's goal was to evaluate the initial concepts of the prototype and was limited in resources. It was conducted as a within-subject study. As a next step, after addressing the issues discussed in the section above, a future study should focus on the long-time use. Also, between-subjects study design should be considered to evaluate unbounded nesting of elements in direct comparison to standard nesting. An opportunity could be software development related projects in university courses like prior studies showed [30]. The potential positive effects of unbounded nesting should be studied more thoroughly. For this purpose, the same prototype but without unbounded nesting could be used in a control group. The initial user study evaluated the concept of the symbolic intelligence and found it to be promising. Future studies should also examine potential

positive effects on managing and scheduling projects in more detail. A control group could use the software without the intelligent feature, relying on manual planning. Generally, more participants should be recruited to raise the statistical expressiveness of the results.

6.2 Areas of improvement for the prototype

In the previous section, the evaluation results of the prototype were discussed, and problems were identified. Based on these findings, areas of improvement are derived and described in the following.

6.2.1 Unbounded nesting

6.2.1.1 Kanban method context

As seen in the previous section, unbounded nesting could provide advantages in visualising a project. However, as noted by a participant, it is unclear how to use the unbounded nesting in the Kanban method context when pulling items into the next column of a Kanban board. When a nested element is being moved into the next column, the relation to its former parent element and the visualisation of this hierarchy is lost. One of the participants suggested an approach to cope with this in the qualitative feedback section. Instead of representing stages within a Kanban board using List elements, Kanban could also be part of the resource concept (see section 3.3.3). The Kanban resource should be configurable and consist of the project's stages. Assigned to an element, it denotes the element's current stage. The view can be switched to a Kanban view showing the elements in their respective stages. In contrast to the current implementation, this Kanban view is provided automatically. If a sub-element's stage differs from its parent elements' stages, the hierarchy to the parents could still be visually indicated; for example, by showing the parent elements in the sub-element's stage with reduced opacity and minimal details.

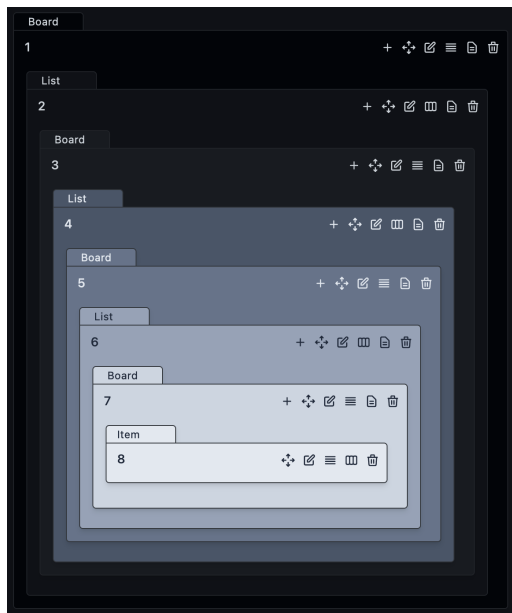
6.2.1.2 Visualisation of unbounded nesting

While the unbounded nesting was perceived positively in the user study and could improve the visualisation of projects, feedback indicates that displaying the unbounded nesting can also clutter the interface. Based on the participant's feedback, two potential improvements for displaying unbounded nesting were identified. One feedback regarded the colouring of elements. The prototype uses different colours for different element types, but, for example, all Board elements are coloured blue. However, this can make the interface cluttered if too many elements of the same type are nested. Therefore, different approaches in colour coding elements should be investigated. One way could be colour coding elements by their level. An example of such a colour-coding can be found in figure 6.1a. A gradient ranging from almost black to almost white is used, getting lighter for deeper nested elements. The approach might allow to easier perceive elements within the same level. Element types could still be perceived through the handle of an element.

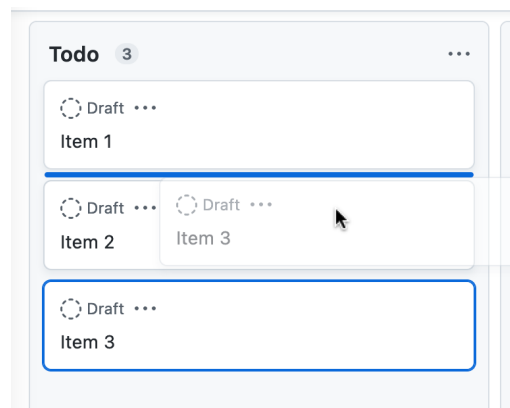
Another way to avoid clutter when displaying the unbounded nesting could be to hide deeper nested elements. In the study, participants proposed two approaches for hiding elements. One technique consists of zoomable interface, showing deeper nested structure only when zooming in. Another option regards collapsible elements and showing child elements only if an element is unfolded.

6.2.1.3 Moving elements in nested situations

Currently, moving elements to different locations must be executed using two different actions, drag and drop or the move button based on the situation. As seen in section 6.1.3, this approach proved to be unintuitive, and participants demanded the possibility to use drag and drop in every move situation. Therefore, efforts should be made to realise unrestricted drag and drop. One of the problems regarding unrestricted drag and drop identified in the conceptional phase (see chapter 3) was indicating the new position of an element. Currently, when dragging elements over other elements, these elements are moved to the side, indicating the new position of the element. Therefore, in deeper nested situations with big elements, elements moving in different directions are confusing. Another way to indicate possible new positions could be by adding a visual indication, as done in the recent version of GitHub Projects [70] (also see section 2.3.2). Figure 6.1b shows a drag and drop interaction in GitHub Projects. The visual indicator could be more practical in a nested context, avoiding moving elements to the side. It would be shown on the destination where the dragged element is currently dragged over. Another problem regarding unrestricted drag and drop was faced in the implementation phase of the prototype, as the utilised drag and drop library did not support it correctly. This issue could be solved using a different drag and drop library such as SortableJS [79].



(a) Elements colour coded by level.



(b) Indication of the new position in GitHub Projects during drag and drop.

Figure 6.1: Areas of improvement for unbounded nesting.

6.2.2 Scheduling feature

This section discusses potential improvements concerning the scheduling feature represented through the symbolic intelligence. Some of the ideas are motivated by feedback gathered in the user study and findings of chapter 2.

6.2.2.1 Visualisation of the scheduling result

One of the areas of improvement identified by the study results concerns the visualisation of the scheduling result.

Timeline view One proposed feature for improving the visualisation was a timeline or calendar view. Currently, the scheduling result is presented through annotating the elements with the position in the execution order as seen in figure 3.6. The temporal details (when a task starts or finishes) are only revealed after hovering over the indication. The timeline view would allow perceiving the temporal details and the execution order of the scheduling at a glance.

Evaluation of the current backlog order Further feedback regarded the existing backlog order (usually the first column of the Kanban board). In the backlog, tasks are often arranged in a prioritised form. Accordingly, its order presents the manually planned execution sequence. Evaluating this order against the proposed order and visualising the result might make comparing both plannings easier. Furthermore, one of the participants demanded the option to accept the proposed order and reorder the backlog accordingly.

Coping with a range of optimal solutions In the prototype, the constraint solver implementation can find an optimal or close-to-optimal solution for the scheduling problem. However, this could include hundreds or even thousands of possible outcoming solutions depending on the problem. Hence, even if the current order of the backlog is already an optimal solution, a different solution might be proposed, potentially confusing the user. This problem could be avoided by calculating the finish date and the number of due dates missed for the current backlog order and comparing this with the proposed solution. If the current order's characteristics are equal or better than the proposed order, this could be indicated to the user, and no different order is proposed.

General improvements Generally, regarding participants' feedback, the visual presentation of the result could be improved. For example, the next task could be highlighted more prominently.

6.2.2.2 Constraint solver enhancements

Other feedback gathered in the study concerned the strategy used by the constraint solver to find a solution. Potential improvements are discussed in the following.

Additional constraints The most proposed improvement for the symbolic intelligence through participants' qualitative feedback was adding more constraints that influence the proposed order. Additional constraints can consider additional aspects within a project for the scheduling, thus improving the scheduling's accuracy. New constraints can be implementable with small effort due to the nature of the constraint solver implementation. Currently, only temporal constraints are used. Other constraints could include the priority of the tasks (derived from resources or the current order of the backlog), dependencies between tasks, WIP limits or other numerical and textual resources.

Support for multiple team members Currently, the implementation of the prototype proposes the task order under the assumption that only one team member works on the project (only one task is processed at a time). In the future, it should be possible to calculate proposals for multiple team members as well. As described in chapter 4 this requires only

small adjustments, as the current implementation of the constraint solver already incorporates the concept of multiple users.

Configurability The symbolic intelligence could also be made more configurable. Firstly, the work schedule, team members working on a project and WIP limits should be adjustable. Also, different strategies and settings for the constraint solver might be selectable. For example, a strategy to cope with due dates that are impossible to fulfil could either move these tasks to the end of the execution sequence or complete them as close to the due dates as possible. Also, users might configure that completing many short tasks is favoured over completing fewer long tasks.

6.2.2.3 Support through data-driven techniques

The current prediction system is solely based on a constraint solver, which finds the optimal task order based on the temporal resources configured by the user beforehand. The informative value of the scheduling result is limited if the resources are not configured or maintained correctly. The accuracy of the scheduling system could be supported through data-driven techniques based on historical project data. Ideas for applying data-driven techniques will be discussed in the following.

Adjust prediction with historical bias One way of improving the prediction might be utilising data of projects' finish dates in the past and how long tasks usually took. Tasks processing durations might have been regularly longer or shorter than the duration estimated by a user beforehand. This information can be used as a factor for calculating the finish date in future estimations. For example, if tasks generally took 20% longer to finish in the past, the assumed duration for calculating the finish date for future tasks can be multiplied by 1.2.

Automatic duration estimation The duration assumed for a task during planning has to be entered manually by the user, making the scheduling result heavily dependent on the accuracy of the user's estimation. As an approach to reduce this dependency, data-driven techniques could be used to calculate the duration automatically. For example, if story points are attached to a task, the velocity of a team or user could be calculated and used to derive the duration of that task. Also, techniques presented in section 2.4 might be promising, such as the deep learning model proposed by Choetkiertikul et al. [17].

Support constraint solver through other approaches Another possibility to enhance the results of the estimation system could be combining it with other approaches, such as the influence diagram proposed by Weflen et al. [51]. The data for weighing the nodes in the diagram could be collected automatically during usage of the application.

6.2.2.4 Scheduling everywhere

Currently, the scheduling can only be triggered on boards containing at least two List sub-elements. This limitation was added as two is the logical minimum of stages within a Kanban board. However, when adapting Kanban as a resource, as discussed in section 6.2.1.1, this limitation could be removed, allowing to trigger the scheduling for every element in the workspace that contains sub-elements.

6.2.3 General usability

The usability of the prototype could still be improved. As seen in section 6.1.3 the most important improvement besides the enhancement of the drag and drop functionality regards the modification of elements. Intuitively this should be possible by clicking the parts of the element that are subject to change, for example, its title.

6.2.4 Additional features

Part of the initial concept of the prototype was to develop an application with essential features. However, important features are still missing for a consumer product. Also, features were demanded by the study's participants. Some of the features will be discussed in the following.

User management and shared boards While different users are available and users can be assigned to tasks using the User resources, no proper user management or shared boards are available. For a consumer product, it should be possible to create user accounts protected by passwords and work collaboratively on workspaces as a team.

WIP limits In order to conform with the Kanban method, it should be possible to configure WIP limits for certain stages.

Searching and filtering elements Another useful feature could be searching and filtering the workspace or composing queries. Such a query could be: "Show all elements with a duration longer than 5 hours."

Additional resources and improved description One feature proposed by participants of the user study is adding additional resources such as images or attachments. Also, it could be helpful to offer formatting capabilities inside the description of an element. A way to achieve this could be integrating the markup language Markdown [87].

Recurring tasks Potential use cases such as studying or laboratory work (see sections 6.3.2 and 6.3.3) include weekly recurring tasks. These could be supported through functionality that automatically adds elements to the backlog, for example, every week. Another option could be functionality for duplicating elements and their sub-elements or creating templates.

Different recommendation systems Currently, the recommendation system consists only of the scheduling system offered by symbolic intelligence. Other types of recommendations could enhance the management of projects even more. Recommendations could be based on statistical data about projects. In the conceptional phase of this work, data of publicly available boards from the public APIs of Trello and GitHub projects was fetched and examined. Although insights of this exploration were not used for the final prototype, which utilised a symbolic intelligence, they could be investigated more in the future. Some interesting ideas for recommendations are presented in the following. In order to evaluate these approaches, larger datasets are required, which could consist of publicly available data and data collected in future studies using our prototype.

- Information on how parent elements are filled and emptied with sub-elements, and how elements move through the stages of a Kanban process:

- Which parent element is just growing but never shrinking?
 - When are specific parent elements filling, when shrinking?
 - In which stage of the process do elements get stuck?
- Information could be used to help identify bottlenecks.
- Information on how elements are created, modified and converted:
 - When do elements get created?
 - What is the probability that an element will be converted to another type of element?
 - Which elements have a large description that also contains bullet points?
- Information could be used for recommending conversions of elements.

6.3 Use cases

As seen in section 2.2.5 the application of the Kanban method is not limited to a specific field. Likewise, the prototype was conceptualised to support a broad range of real-life use cases. The following section selects three examples of potential use cases and explains them briefly.

6.3.1 Agile software development

Agile software development presents the standard use case of a Kanban board-based tool. Here, the enhancements of our prototype might be useful. The stages of a software development project's process can be represented using the Kanban method and its board. The unbounded nesting could be used to better visualise different hierarchies in the project, showing larger project scopes and sub-tasks and thus giving a better overview of the project. The symbolic intelligence could be used as a forecasting system for evaluating project deadlines. Also, if not clear which features should be developed first to reach all deadlines, the task order proposition might be useful.

6.3.2 Studying at a university

A semester at a university usually consists of attending multiple courses with different exercises, deadlines and exams. It presents an ideal opportunity to be managed using our Kanban board-based application. Different ways of representing a semester are imaginable and supported through the application's flexible nesting capabilities. For example, it could be represented using a single Kanban board with the standard stages, *open*, *in progress*, and *done*. Courses could be represented using Lists containing sub-Lists for the different types within the course, such as weekly exercises, full-semester projects and exams. Each of these Lists can then contain Items representing weekly exercise sheets. Temporal resources can represent temporal constraints of exercise sheets like start date and a due date. For weekly tasks, a recurring task feature as explained in section 6.2.4 might be useful. Another way of representing the semester can be multiple Kanban boards representing different types of tasks such as weekly exercises and exams. The stages of these boards could then be more detailed and include, for example, *exercise sheet in correction* and *exercise sheet graded*. The symbolic intelligence could assist in managing a week with different due dates for different exercises. Applying it in the context of a longer timespan, such as a whole semester or the exam phase, could indicate if too many courses are taken and if succeeding in finishing all these courses is realistic.

6.3.3 Laboratory work organisation

Organising work in a laboratory, such as analysing COVID-19 PCR test samples, can be another potential use case. A week or longer timespan could be represented through a single Kanban board or multiple Kanban boards representing different work areas. A week in the laboratory consists of many tasks with predefined time periods. Certain tasks depend on start dates, such as the new samples arriving or a colleague coming to work at a specific time. For others, due dates are relevant, such as probes or reports that must be delivered to other departments until a certain time. In such a scenario, especially the scheduling system could be useful. It could help find an order of tasks to meet all temporal constraints and give an overview of whether the different tasks are still on time. As many tasks are repeated regularly, a helpful addition could be the recurring tasks feature described in section 6.2.4.


CHAPTER 7

Conclusion


This work presented a Kanban board-based web application prototype featuring unbounded nesting of Boards, Lists and Items. Furthermore, a recommendation system to support the scheduling of projects has been implemented using a symbolic intelligence based on a constraint solver. Based on pre-configured temporal constraints, the symbolic intelligence proposes a task order for finishing a project as early as possible and minimising the number of missed due dates. The prototype and its concepts were evaluated in a user study with 13 participants. For visualising an example project, participants used a mean maximum nesting depth of 3.08 and favoured arrangements with a mean maximum nesting depth of 3.77. Both values were significantly greater than the current standard depth of 2. Furthermore, participants perceived unbounded nesting positively. The results indicate that unbounded nesting can be a useful enhancement of Kanban board-based tools for visualising projects. Furthermore, in an example project with ten tasks, the participant's manually planned task execution order would have led to a significantly later project finish date with more due dates missed than the execution order proposed by the symbolic intelligence. Also, participants perceived the symbolic intelligence positively and welcomed the feature. The results indicate that users have problems correctly estimating the temporal constraints of larger projects. Therefore, the symbolic intelligence can support project scheduling in Kanban board-based tools. Overall, the usability of the prototype was also perceived positively. Additionally, the user study revealed areas of improvement for the prototype regarding unbounded nesting, symbolic intelligence and usability. These should be addressed in future iterations of the prototype. The study focused on initially evaluating the concepts of the prototype. Therefore, the result's expressiveness regarding long time use is limited. For unbounded nesting, only the visualisation of a project was examined. Future work should investigate the application of unbounded nesting during long time use and when working with the Kanban method. Data-driven approaches for the recommendation system should be further investigated to improve its expressiveness and reduce its dependency on prior user input. This work investigated novel approaches for Kanban board-based tools and validated their potential. The concepts and prototype implementation presented in this work can be used for future Kanban board-based implementations and further investigations of the novel approaches. Finally, unbounded nesting and a symbolic intelligence were introduced to improve board-based tools without comprising simplicity. Therefore, this work makes a small contribution to helping people manage their tasks and projects in different application areas.

Source code


Backend

-  <https://github.com/tmrdlt/masterthesis-khipu>
- Backend of the prototype written in the Scala Programming Language.

Frontend

-  <https://github.com/tmrdlt/masterthesis-khipu-frontend>
- Frontend of the prototype written in TypeScript.

Evaluation and visualisation

-  <https://github.com/tmrdlt/masterthesis-evaluation>
- Statistic evaluation and visualisation of the user study results using R.
- Additional calculations using Python and JavaScript.
- Architecture and class diagram of prototype.

Extended NASA-TLX questionnaire

Task Questionnaire

Click on each scale at the point that best indicates your experience of the task

<p style="text-align: center;">Mental Demand</p> <div style="border: 1px solid black; height: 20px; width: 100%; position: relative;"> <div style="position: absolute; left: 0; bottom: 0;">Low</div> <div style="position: absolute; right: 0; bottom: 0;">High</div> </div>	<p>How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc)? Was the task easy or demanding, simple or complex, exacting or forgiving?</p>
<p style="text-align: center;">Physical Demand</p> <div style="border: 1px solid black; height: 20px; width: 100%; position: relative;"> <div style="position: absolute; left: 0; bottom: 0;">Low</div> <div style="position: absolute; right: 0; bottom: 0;">High</div> </div>	<p>How much physical activity was required (e.g. pushing, pulling, turning, controlling, activating, etc)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?</p>
<p style="text-align: center;">Temporal Demand</p> <div style="border: 1px solid black; height: 20px; width: 100%; position: relative;"> <div style="position: absolute; left: 0; bottom: 0;">Low</div> <div style="position: absolute; right: 0; bottom: 0;">High</div> </div>	<p>How much time pressure did you feel due to the rate of pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?</p>
<p style="text-align: center;">Performance</p> <div style="border: 1px solid black; height: 20px; width: 100%; position: relative;"> <div style="position: absolute; left: 0; bottom: 0;">Poor</div> <div style="position: absolute; right: 0; bottom: 0;">Good</div> </div>	<p>How successful do you think you were in accomplishing the goals of the tasks? How satisfied were you with your performance in accomplishing these goals?</p>
<p style="text-align: center;">Effort</p> <div style="border: 1px solid black; height: 20px; width: 100%; position: relative;"> <div style="position: absolute; left: 0; bottom: 0;">Low</div> <div style="position: absolute; right: 0; bottom: 0;">High</div> </div>	<p>How hard did you have to work (mentally and physically) to accomplish your level of performance?</p>
<p style="text-align: center;">Frustration</p> <div style="border: 1px solid black; height: 20px; width: 100%; position: relative;"> <div style="position: absolute; left: 0; bottom: 0;">Low</div> <div style="position: absolute; right: 0; bottom: 0;">High</div> </div>	<p>How insecure, discouraged, irritated, stressed and annoyed versus secure, gratified, content, relaxed and complacent did you feel during the tasks?</p>
<p style="text-align: center;">Control</p> <div style="border: 1px solid black; height: 20px; width: 100%; position: relative;"> <div style="position: absolute; left: 0; bottom: 0;">Low</div> <div style="position: absolute; right: 0; bottom: 0;">High</div> </div>	<p>How much in control did you feel during the tasks?</p>
<p style="text-align: center;">Exploration</p> <div style="border: 1px solid black; height: 20px; width: 100%; position: relative;"> <div style="position: absolute; left: 0; bottom: 0;">Low</div> <div style="position: absolute; right: 0; bottom: 0;">High</div> </div>	<p>How much did you feel encouraged to explore the interface and/or different ways of solving the tasks?</p>
<p style="text-align: center;">Motivation</p> <div style="border: 1px solid black; height: 20px; width: 100%; position: relative;"> <div style="position: absolute; left: 0; bottom: 0;">Low</div> <div style="position: absolute; right: 0; bottom: 0;">High</div> </div>	<p>How motivated by the interface did you feel to solve the tasks?</p>
<p style="text-align: center;">Enjoyment</p> <div style="border: 1px solid black; height: 20px; width: 100%; position: relative;"> <div style="position: absolute; left: 0; bottom: 0;">Low</div> <div style="position: absolute; right: 0; bottom: 0;">High</div> </div>	<p>How much in did you enjoy using the interface?</p>

Continue >>

Quantitative user study data (1)

Intro				Task 1.1									Max depth personal
Participant ID	Age	Have you used board-based tools before?	Prior experience	Duration	Total events	CONVERT	CREATE	DELETE	MOVE	REORDER	UPDATE	UPDATE_RESOURCES	
2	27	yes	6	04:40	40	5	8	1	19	4	3	0	4
3	28	yes	7	09:50	51	2	7	2	24	14	2	0	3
4	28	yes	7	04:56	31	2	0	2	14	11	2	0	2
5	27	yes	7	09:50	35	5	10	1	19	0	0	0	2
6	28	no		08:41	24	0	2	0	12	6	4	0	2
7	25	yes	3	09:43	51	13	3	0	28	7	0	0	2
8	26	yes	6	06:01	22	3	0	0	9	9	0	1	3
9	30	yes	2	07:39	46	0	4	0	18	24	0	0	3
10	27	yes	7	09:08	64	3	10	4	36	9	0	2	4
11	26	yes	7	05:09	23	4	3	0	15	1	0	0	3
12	29	no		09:29	47	12	0	0	20	15	0	0	5
13	27	yes	7	05:50	32	1	5	0	17	8	1	0	4
14	28	no		06:00	25	4	0	0	11	10	0	0	3

Task 1.1: Extended NASA-TLX													
Participant ID	Mental Demand	Physical Demand	Temporal Demand	Performance	Effort	Frustration	Control	Exploration	Motivation	Enjoyment	Workload	Sense of control	Creativity Support Index
2	60	5	5	60	35	5	85	60	35	25	27,5	85	40
3	30	20	30	70	50	15	90	80	65	65	33,33333333	90	70
4	75	15	20	80	35	30	80	90	85	80	37,5	80	85
5	80	35	80	65	75	15	50	75	80	75	55,83333333	50	76,66666667
6	75	50	30	25	55	70	20	15	70	70	39,16666667	20	51,66666667
7	70	25	15	80	40	10	90	90	90	80	38,33333333	90	86,66666667
8	65	15	25	90	15	5	95	80	70	75	35	95	75
9	65	25	5	90	25	5	85	95	70	90	35	85	85
10	75	65	45	75	50	20	75	95	95	90	51,66666667	75	93,33333333
11	50	70	60	65	30	25	70	85	70	70	45,83333333	70	75
12	80	50	80	35	70	85	60	75	45	70	52,5	60	63,33333333
13	70	35	60	70	55	35	100	80	70	60	48,33333333	100	70
14	40	10	15	80	25	20	85	80	90	90	28,33333333	85	86,66666667

Task 1.2			
Participant ID	Max depth preferred	Better than personal solution?	Max depth overall
2	2	yes	2
3	5	yes	5
4	3	no	2
5	3	yes	3
6	5	yes	5
7	3	yes	3
8	2	yes	2
9	5	yes	5
10	3	no	4
11	5	yes	5
12	3	yes	3
13	5	yes	5
14	5	yes	5

Quantitative user study data (2)

69

Task 2.1												
Participant ID	Duration	Finish date est	Finish date real	Diff h est vs real	Diff h real vs opt	Failed dues est	Failed dues real	Diff dues est vs real	Diff dues real vs opt	Order diff to prop	Order diff to opt	
2	03:00	01.11.2021 18:00	01.11.2021 18:00	0	0	0	0	0	0	0,8	1	
3	03:00	01.11.2021 18:00	01.11.2021 18:00	0	0	0	1	1	1	0,8	0,8	
4	01:30	01.11.2021 18:00	01.11.2021 18:00	0	0	0	0	0	0	0,8	1	
5	03:00	01.11.2021 18:00	01.11.2021 18:00	0	0	0	0	0	0	0,8	1	
6	03:00	01.11.2021 18:00	02.11.2021 11:00	1	1	0	1	1	1	0,6	0,8	
7	03:00	01.11.2021 18:00	01.11.2021 18:00	0	0	0	0	0	0	0,8	1	
8	03:00	01.11.2021 18:00	01.11.2021 18:00	0	0	0	0	0	0	0,8	1	
9	01:50	01.11.2021 16:30	01.11.2021 18:00	0	0	0	1	1	1	0,6	0,8	
10	01:58	01.11.2021 18:00	01.11.2021 18:00	0	0	0	0	0	0	0,8	1	
11	01:26	02.11.2021 10:30	02.11.2021 10:30	0	0,5	0	1	1	1	0,6	0,6	
12	01:50	01.11.2021 18:00	01.11.2021 18:00	0	0	0	0	0	0	0,8	1	
13	01:10	01.11.2021 18:00	01.11.2021 18:00	0	0	0	0	0	0	0,8	1	
14	03:00	01.11.2021 18:00	02.11.2021 10:30	0,5	0,5	1	1	0	1	0,2	0,6	

Task 2.2												
Participant ID	Duration	Finish date est	Finish date real	Diff h est vs real	Diff h real vs opt	Failed dues est	Failed dues real	Diff dues est vs real	Diff dues real vs opt	Order diff to prop	Order diff to opt	
2	06:00	05.11.2021 12:00	08.11.2021 16:00	12	6	2	2	0	0	0,4	0,9	
3	06:00	05.11.2021 18:00	08.11.2021 16:00	6	6	0	3	3	1	0,3	0,9	
4	05:28	04.11.2021 12:00	09.11.2021 12:00	24	10	2	2	0	0	0,2	0,8	
5	06:00	04.11.2021 18:00	09.11.2021 11:00	17	9	1	3	2	1	0,5	0,8	
6	05:21	05.11.2021 18:00	08.11.2021 16:00	6	6	1	3	2	1	0,3	0,9	
7	06:00	05.11.2021 17:00	05.11.2021 18:00	1	0	1	3	2	1	0,7	0,9	
8	06:00	06.11.2021 10:00	10.11.2021 12:00	18	18	1	3	2	1	0,4	0,8	
9	06:00	05.11.2021 18:00	05.11.2021 18:00	0	0	1	4	3	2	0,3	0,8	
10	05:37	05.11.2021 18:00	08.11.2021 13:00	3	3	2	2	0	0	0,5	0,9	
11	04:33	08.11.2021 16:00	08.11.2021 16:00	8	6	2	3	1	1	0,3	0,9	
12	05:30	05.11.2021 15:00	08.11.2021 12:00	5	2	1	3	2	1	0,3	0,8	
13	05:30	05.11.2021 18:00	05.11.2021 18:00	0	0	3	2	1	0	0,4	1	
14	06:00	05.11.2021 18:00	08.11.2021 16:00	6	6	4	3	1	1	0,3	0,9	

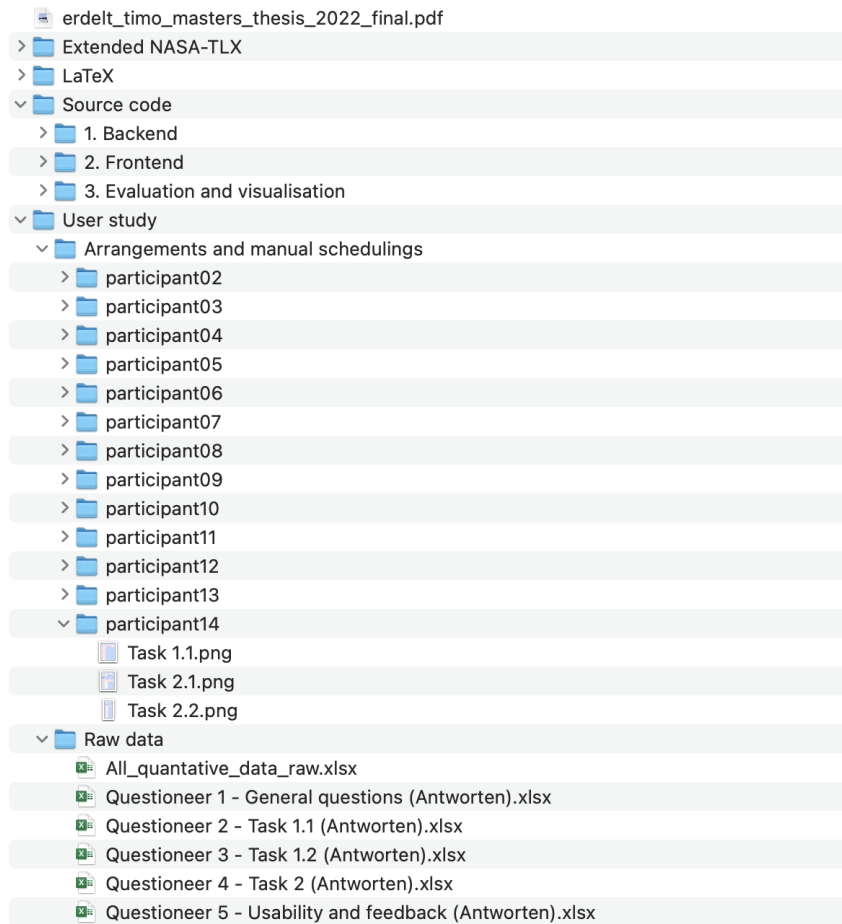
Likert-type questions													
Participant ID	Task 1.1			Task 2							Outro		
	LTQ01	LTQ02	LTQ03	LTQ04	LTQ05	LTQ06	LTQ07	LTQ08	LTQ09	LTQ10	LTQ11	LTQ12	LTQ13
2	5	5	4	1	5	5	5	5	2	5	5	5	5
3	5	4	4	1	3	2	4	4	5	5	4	4	5
4	5	5	5	1	3	4	4	5	5	5	5	5	5
5	5	5	4	1	5	5	5	5	4	5	4	4	5
6	4	4	4	3	3	4	5	3	3	2	3	3	5
7	5	5	4	1	4	5	4	5	4	4	5	4	5
8	5	4	5	1	2	3	1	4	2	4	5	5	4
9	5	5	5	1	5	5	5	5	5	5	5	5	5
10	4	4	4	1	4	3	5	5	4	5	4	4	5
11	5	4	3	1	4	5	4	5	4	5	4	3	5
12	4	4	3	2	4	5	4	5	2	4	4	5	5
13	5	5	4	1	5	5	4	5	4	5	4	4	5
14	5	5	4	1	4	5	5	5	5	5	5	5	4

Appendix

Quantitative user study data (3)

Code	Part	Text
LTQ01	Task 1.1	I understood what my task was about.
LTQ02	Task 1.1	The software supported me visualizing the project.
LTQ03	Task 1.1	The arrangement I created matches my mental image of the project.
LTQ04	Task 2	I feel patronized by the software making the planning for me.
LTQ05	Task 2	I find the proposal easy to perceive.
LTQ06	Task 2	I find the proposal intelligible.
LTQ07	Task 2	I trust the proposal to be better than my manual planning.
LTQ08	Task 2	I would find an automatic planning feature like this to be helpful.
LTQ09	Task 2	It is likely that I would use an automatic planning feature like this.
LTQ10	Task 2	If my privacy is guaranteed, I am generally open to intelligent features in a software like this.
LTQ11	Outro	I found the interface of the software easy to understand.
LTQ12	Outro	The features of the software were clear to me. I understand what can be done using the features.
LTQ13	Outro	I liked the possibility to arrange elements (boards, lists, items) in an arbitrarily nested way.

Content of attached ZIP file



References

- [1] Pekka Abrahamsson, ed. *Agile software development methods: review and analysis*. eng. VTT publications 478. Espoo, Finland: VTT, 2002. ISBN: 9789513860097.
- [2] Muhammad Ovais Ahmad, Denis Dennehy, Kieran Conboy, and Markku Oivo. "Kanban in software engineering: A systematic mapping study". en. In: *Journal of Systems and Software* 137 (Mar. 2018), pp. 96–113. ISSN: 01641212. DOI: 10.1016/j.jss.2017.11.045.
- [3] Muhammad Ovais Ahmad, Jouni Markkula, and Markku Oivo. "Kanban in Software Development: A Systematic Literature Review". In: *Proceedings of the 2013 39th Euromicro Conference on Software Engineering and Advanced Applications*. SEAA '13. USA: IEEE Computer Society, 2013, pp. 9–16. ISBN: 9780769550916. DOI: 10.1109/SEAA.2013.28.
- [4] Muhammad Ovais Ahmad, Jouni Markkula, Markku Oivo, and Pasi Kuvaja. "Usage of Kanban in Software Companies An empirical study on motivation, benefits and challenges". en. In: (2014). DOI: 10.13140/2.1.5145.1849.
- [5] Hamzah Alaidaros, Mazni Omar, and Rohaida Romli. "Towards an improved software project monitoring task model of Agile Kanban method". en. In: *International Journal of Supply Chain Management (IJSCM)* 7.3 (June 2018), pp. 118–125. ISSN: 2050-7399. URL: <https://web.archive.org/web/20220216085957/http://ojs.excelingtech.co.uk/index.php/IJSCM/article/view/2250> (visited on 02/16/2022).
- [6] Hamzah Alaidaros, Mazni Omar, and Rohaida Romli. "The state of the art of agile kanban method: challenges and opportunities". In: *Independent Journal of Management & Production* 12.8 (Dec. 2021), pp. 2535–2550. ISSN: 2236-269X, 2236-269X. DOI: 10.14807/ijmp.v12i8.1482.
- [7] Hamzah Alaidaros, Mazni Omar, Rohaida Romli, and Adnan Hussein. "The Development and Evaluation of A Progress Monitoring Prototype Tool for Software Project Management". In: *2019 First International Conference of Intelligent Computing and Engineering (ICOICE)*. Hadhramout, Yemen: IEEE, Dec. 2019, pp. 1–9. ISBN: 9781728144870. DOI: 10.1109/ICOICE48418.2019.9035146.
- [8] Bashaer Alsaadi and Kawther Saeedi. "Data-driven effort estimation techniques of agile user stories: a systematic literature review". en. In: *Artificial Intelligence Review* (Jan. 2022). ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-021-10132-x.

- [9] David J. Anderson. *Kanban: successful evolutionary change for your technology business*. eng. Sequim, Washington, USA: Blue Hole Press, 2010. ISBN: 9780984521401.
- [10] Renae Aurisch, Mohiuddin Ahmed, and Abu Barkat. "An outlook at Agile methodologies for the independent games developer". en. In: *International Journal of Computers and Applications* 43.8 (Sept. 2021), pp. 812–818. ISSN: 1206-212X, 1925-7074. DOI: 10.1080/1206212X.2019.1621463.
- [11] Julian M. Bass. "Improving writing processes using lean and Kanban". en. In: *Learned Publishing* 29.4 (2016), pp. 307–310. ISSN: 1741-4857. DOI: 10.1002/leap.1045.
- [12] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Marick Brian, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. "Manifesto for Agile Software Development". In: (2001). URL: <https://web.archive.org/web/20220216090044/https://agilemanifesto.org/> (visited on 02/16/2022).
- [13] Jim Benson and Tonianne De Maria Barry. *Personal Kanban: mapping work, navigating life*. eng. First edition. Seattle, Washington, USA: Modus Cooperandi Press, 2011. ISBN: 9781453802267.
- [14] Harry N Boone and Deborah A Boone. "Analyzing likert data". In: *Journal of extension* 50.2 (2012), pp. 1–5. URL: <https://web.archive.org/web/20220216090136/https://archives.joe.org/joe/2012april/tt2.php> (visited on 02/16/2022).
- [15] Eric Brechner and James Waletzky. *Agile project management with Kanban*. Best practices. OCLC: ocn906825505. Redmond, Washington, USA: Microsoft Press, 2015. ISBN: 9780735698956.
- [16] Erin Cherry and Celine Latulipe. "Quantifying the Creativity Support of Digital Tools through the Creativity Support Index". In: *ACM Transactions on Computer-Human Interaction* 21.4 (June 2014), 21:1–21:25. ISSN: 1073-0516. DOI: 10.1145/2617588.
- [17] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. "A Deep Learning Model for Estimating Story Points". In: *IEEE Transactions on Software Engineering* 45.7 (July 2019), pp. 637–656. ISSN: 0098-5589, 1939-3520, 2326-3881. DOI: 10.1109/TSE.2018.2792473.
- [18] Denis Dennehy and Kieran Conboy. "Identifying Challenges and a Research Agenda for Flow in Software Project Management". en. In: *Project Management Journal* 49.6 (Dec. 2018), pp. 103–118. ISSN: 8756-9728, 1938-9507. DOI: 10.1177/8756972818800559.
- [19] Mia Y. Dong, Kristian Sandberg, Bo M. Bibby, Michael N. Pedersen, and Morten Overgaard. "The development of a sense of control scale". In: *Frontiers in Psychology* 6 (2015), p. 1733. ISSN: 1664-1078. DOI: 10.3389/fpsyg.2015.01733.
- [20] Scott Downey and Jeff Sutherland. "Scrum Metrics for Hyperproductive Teams: How They Fly like Fighter Aircraft". In: *Proceedings of the 2013 46th Hawaii International Conference on System Sciences*. HICSS '13. USA: IEEE Computer Society, Jan. 2013, pp. 4870–4878. ISBN: 9780769548920. DOI: 10.1109/HICSS.2013.471.

- [21] M Fitriawati and R H Lestari. "Design of the Information System for Kindergarten Learning Evaluation used Kanban Methodology". In: *IOP Conference Series: Materials Science and Engineering* 662.2 (Nov. 2019), p. 022025. ISSN: 1757-8981, 1757-899X. DOI: 10.1088/1757-899X/662/2/022025.
- [22] Sandra G. Hart. "Nasa-Task Load Index (NASA-TLX); 20 Years Later". en. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50.9 (Oct. 2006), pp. 904–908. ISSN: 2169-5067. DOI: 10.1177/154193120605000909.
- [23] Enric Senabre Hidalgo. "Adapting the scrum framework for agile project management in science: case study of a distributed research initiative". en. In: *Heliyon* 5.3 (Mar. 2019), e01447. ISSN: 24058440. DOI: 10.1016/j.heliyon.2019.e01447.
- [24] Heather A. Johnson. "Trello". In: *Journal of the Medical Library Association : JMLA* 105.2 (Apr. 2017), pp. 209–211. ISSN: 1536-5050. DOI: 10.5195/jmla.2016.49.
- [25] Henrik Kniberg and Mattias Skarin. *Kanban and Scrum: making the most of both*. eng. InfoQ enterprise software development series. C4Media, 2010. ISBN: 9780557138326. URL: <https://web.archive.org/web/20220216090220/https://www.infoq.com/minibooks/kanban-scrum-minibook/> (visited on 02/16/2022).
- [26] C. Larman and V.R. Basili. "Iterative and incremental developments. a brief history". In: *Computer* 36.6 (June 2003), pp. 47–56. ISSN: 0018-9162. DOI: 10.1109/MC.2003.1204375.
- [27] Howard Lei, Farnaz Ganjeizadeh, Pradeep Kumar Jayachandran, and Pinar Ozcan. "A statistical analysis of the effects of Scrum and Kanban on software development projects". en. In: *Robotics and Computer-Integrated Manufacturing* 43 (Feb. 2017), pp. 59–67. ISSN: 07365845. DOI: 10.1016/j.rcim.2015.12.001.
- [28] Jeffrey K Liker. *The Toyota way: 14 management principles from the world's greatest manufacturer*. New York, USA: McGraw-Hill Education, 2004. ISBN: 9780071392310.
- [29] Viljan Mahnič and Tomaž Hovelja. "On using planning poker for estimating user stories". en. In: *Journal of Systems and Software* 85.9 (Sept. 2012), pp. 2086–2095. ISSN: 01641212. DOI: 10.1016/j.jss.2012.04.005.
- [30] Christoph Matthies. "Scrum2kanban: integrating kanban and scrum in a university software engineering capstone course". en. In: *Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials*. Gothenburg Sweden: ACM, June 2018, pp. 48–55. ISBN: 9781450357500. DOI: 10.1145/3194779.3194784.
- [31] Subhas Misra, Vinod Kumar, Uma Kumar, Kamel Fantazy, and Mahmud Akhter. "Agile software development practices: evolution, principles, and criticisms". en. In: *International Journal of Quality & Reliability Management* 29.9 (Oct. 2012), pp. 972–980. ISSN: 0265-671X. DOI: 10.1108/02656711211272863.
- [32] Shun Nakazawa, Kazuki Komatsu, Tetsuo Tanaka, and Kazunori Matsumoto. "Development and Evaluation of Large-Screen Digital Kanban with Smartphone Operation". In: *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*. Hamamatsu: IEEE, July 2017, pp. 295–300. ISBN: 9781538606216. DOI: 10.1109/IIAI-AAI.2017.151.

- [33] Shun Nakazawa and Tetsuo Tanaka. "Prototype of Kanban Tool and Preliminary Evaluation of Visualizing Method for Task Assignment". In: *2015 International Conference on Computer Application Technologies*. Matsue, Japan: IEEE Computer Society, Aug. 2015, pp. 48–49.
DOI: 10.1109/CCATS.2015.21.
- [34] Shun Nakazawa and Tetsuo Tanaka. "Development and Application of Kanban Tool Visualizing the Work in Progress". In: *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*. Kumamoto, Japan: IEEE, July 2016, pp. 908–913. ISBN: 9781467389853.
DOI: 10.1109/IIAI-AAI.2016.156.
- [35] Donald A. Norman. *The design of everyday things*. Revised and expanded edition. New York, New York: Basic Books, 2013. ISBN: 9780465050659.
- [36] Hong Yeow Ong, Chen Wang, and Nurshuhada Zainon. "Integrated Earned Value Gantt Chart (EV-Gantt) Tool for Project Portfolio Planning and Monitoring Optimization". en. In: *Engineering Management Journal* 28.1 (Jan. 2016), pp. 39–53. ISSN: 1042-9247, 2377-0643.
DOI: 10.1080/10429247.2015.1135033.
- [37] Julia Paredes, Craig Anslow, and Frank Maurer. "Information Visualization for Agile Software Development". In: *Proceedings of the 2014 Second IEEE Working Conference on Software Visualization*. VISSOFT '14. USA: IEEE Computer Society, 2014, pp. 157–166. ISBN: 9781479961504.
DOI: 10.1109/VISSOFT.2014.32.
- [38] David L. Parnas and Paul C. Clements. "A rational design process: How and why to fake it". en. In: *Formal Methods and Software Development*. Ed. by G. Goos, J. Hartmanis, D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, H. Stoer, N. Wirth, Hartmut Ehrig, Christiane Floyd, Maurice Nivat, and James Thatcher. Vol. 186. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 80–100.
DOI: 10.1007/3-540-15199-0_6.
- [39] Mary Poppendieck and Tom Poppendieck. *Lean software development: an agile toolkit*. The agile software development series. Boston, Massachusetts, USA: Addison-Wesley, 2003. ISBN: 9780321150783.
- [40] Florian Raith, Ingo Richter, and Robert Lindermeier. "How Project-management-tools are used in Agile Practice: Benefits, Drawbacks and Potentials". en. In: *Proceedings of the 21st International Database Engineering & Applications Symposium on - IDEAS 2017*. Bristol, United Kingdom: ACM Press, 2017, pp. 30–39. ISBN: 9781450352208.
DOI: 10.1145/3105831.3105865.
- [41] John W. Ratclif and David Metzener. *Pattern Matching: the Gestalt Approach*. July 1988. URL: <https://web.archive.org/web/20200209031652/https://www.drdobbs.com/database/pattern-matching-the-gestalt-approach/184407970?pgno=5> (visited on 02/09/2020).
- [42] W. W. Royce. "Managing the development of large software systems: concepts and techniques". In: *Proceedings of the 9th international conference on Software Engineering*. ICSE '87. Washington, DC, USA: IEEE Computer Society Press, Mar. 1987, pp. 328–338. ISBN: 9780897912167.
DOI: 10.5555/41765.41801.

- [43] Jullia Saad, Priscila Portela Costa, Alexandre Alvaro, and Luciana A. M. Zaina. "Developers experience (DX) in ALM software tools: an investigation on virtual kanban boards". In: *Proceedings of the 34th Brazilian Symposium on Software Engineering*. SBES '20. Natal, Brazil: Association for Computing Machinery, Oct. 2020, pp. 167–172. ISBN: 9781450387538.
DOI: 10.1145/3422392.3422475.
- [44] Jeffrey Saltz, Ivan Shamshurin, and Kevin Crowston. "Comparing Data Science Project Management Methodologies via a Controlled Experiment". In: 2017.
DOI: 10.24251/HICSS.2017.120.
- [45] Jeffrey S. Saltz. "Visualizing Kanban Work: Towards an Individual Contributor View". In: *AMCIS 2019 Proceedings* (July 2019).
URL: https://web.archive.org/web/20220216085836/https://aisel.aisnet.org/amcis2019/it_project_mgmt/it_project_mgmt/7/ (visited on 02/16/2022).
- [46] Mahrukh Sameen Mirza and Soma Datta. "Strengths and Weakness of Traditional and Agile Processes - A Systematic Review". In: *Journal of Software* 14.5 (May 2019), pp. 209–219. ISSN: 1796217X.
DOI: 10.17706/jsw.14.5.209-219.
- [47] Sirikorn Santirojanakul. "The development of sports science knowledge management systems through CommonKADS and digital Kanban board". In: *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*. Penang: IEEE, Apr. 2018, pp. 119–124. ISBN: 9781538635278.
DOI: 10.1109/ISCAIE.2018.8405455.
- [48] Saad Shafiq, Irum Inayat, and Muhammad Abbas. "Communication Patterns of Kanban Teams and Their Impact on Iteration Performance and Quality". In: *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Kallithea-Chalkidiki, Greece: IEEE, Aug. 2019, pp. 164–168. ISBN: 9781728134215.
DOI: 10.1109/SEAA.2019.00033.
- [49] Renata Souza, Larissa Rocha, Franklin Silva, and Ivan Machado. "Investigating Agile Practices in Software Startups". en. In: *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*. Salvador Brazil: ACM, Sept. 2019, pp. 317–321. ISBN: 9781450376518.
DOI: 10.1145/3350768.3350786.
- [50] Adam Trendowicz and Ross Jeffery. *Software project effort estimation: foundations and best practice guidelines for success*. eng. Cham, Switzerland: Springer, 2014. ISBN: 978-3319036298.
DOI: 10.1007/978-3-319-03629-8.
- [51] Eric Weflen, Cameron A. MacKenzie, and Iris V. Rivero. "An influence diagram approach to automating lead time estimation in Agile Kanban project management". en. In: *Expert Systems with Applications* 187 (Jan. 2022), p. 115866. ISSN: 0957-4174.
DOI: 10.1016/j.eswa.2021.115866.
- [52] J P Womack and D T Jones. "Lean Thinking—Banish Waste and Create Wealth in your Corporation". en. In: *Journal of the Operational Research Society* 48.11 (Nov. 1997), pp. 1148–1148. ISSN: 0160-5682, 1476-9360.
DOI: 10.1057/palgrave.jors.2600967.
- [53] Wael Zayat and Ozlem Senvar. "Framework Study for Agile Software Development Via Scrum and Kanban". en. In: *International Journal of Innovation and Technology Management* 17.04 (June 2020), p. 2030002. ISSN: 0219-8770, 1793-6950.
DOI: 10.1142/S0219877020300025.

Web References

- [54] Atlassian. *Jira Software*.
URL: <https://web.archive.org/web/20220216084334/https://www.atlassian.com/software/jira>
(visited on 02/16/2022).
- [55] Atlassian. *The Trello REST API*.
URL: <https://web.archive.org/web/20220216083006/https://developer.atlassian.com/cloud/trello/rest/api-group-actions/>
(visited on 02/16/2022).
- [56] Atlassian. *Trello*.
URL: <https://web.archive.org/web/20220216085110/https://trello.com/home>
(visited on 02/16/2022).
- [57] Digital.ai. *15th State Of Agile Report*. July 2021.
URL: <https://web.archive.org/web/20220216084141/https://digital.ai/resource-center/analyst-reports/state-of-agile-report>
(visited on 02/16/2022).
- [58] Python Software Foundation. *difflib — Helpers for computing deltas*.
URL: <https://web.archive.org/web/20220216084722/https://docs.python.org/3/library/difflib.html>
(visited on 02/16/2022).
- [59] PostgreSQL Global Development Group. *PostgreSQL*.
URL: <https://web.archive.org/web/20220216084654/https://www.postgresql.org/>
(visited on 02/16/2022).
- [60] Red Hat. *OptaPlanner*.
URL: <https://web.archive.org/web/20220216084606/https://www.optaplanner.org/>
(visited on 02/16/2022).
- [61] Asana Inc. *10-K anual report*.
URL: <https://web.archive.org/web/20220216090841/https://dl8rn0p25nwr6d.cloudfront.net/CIK-0001477720/38df8983-89d8-458e-8b75-601bddbb9361.pdf>
(visited on 02/16/2022).

- [62] Asana Inc. *Asana*.
URL: <https://web.archive.org/web/20220216083915/https://asana.com/>
(visited on 02/16/2022).
- [63] Lightbend Inc. *Akka HTTP*.
URL: <https://web.archive.org/web/20220216083824/https://doc.akka.io/docs/akka-http/current/>
(visited on 02/16/2022).
- [64] Meta Platforms Inc. *React*.
URL: <https://web.archive.org/web/20220216084816/https://reactjs.org/>
(visited on 02/16/2022).
- [65] JetBrains. *YouTrack*.
URL: <https://web.archive.org/web/20220216085711/https://www.jetbrains.com/youtrack/>
(visited on 02/16/2022).
- [66] Tobias Koppers. *Webpack*.
URL: <https://web.archive.org/web/20220216085158/https://webpack.js.org/>
(visited on 02/16/2022).
- [67] Tailwind Labs. *Tailwind CSS*.
URL: <https://web.archive.org/web/20220216085044/https://tailwindcss.com/>
(visited on 02/16/2022).
- [68] Sebastian McKenzie. *Babel*.
URL: <https://web.archive.org/web/20220216084032/https://babeljs.io/>
(visited on 02/16/2022).
- [69] Microsoft. *GitHub*.
URL: <https://web.archive.org/web/20220216084224/https://github.com/>
(visited on 02/16/2022).
- [70] Microsoft. *GitHub Projects*.
URL: <https://web.archive.org/web/20220216084300/https://github.com/features/issues>
(visited on 02/16/2022).
- [71] Microsoft. *GitHub REST API*.
URL: <https://web.archive.org/web/20220216083217/https://docs.github.com/en/rest>
(visited on 02/16/2022).
- [72] Microsoft. *Microsoft Teams*.
URL: <https://web.archive.org/web/20220216084405/https://www.microsoft.com/en-gb/microsoft-teams/group-chat-software>
(visited on 02/16/2022).
- [73] Microsoft. *TypeScript*.
URL: <https://web.archive.org/web/20220216085131/https://www.typescriptlang.org/>
(visited on 02/16/2022).

- [74] Miro. *Miro*.
URL: <https://web.archive.org/web/20220216084436/https://miro.com/>
(visited on 02/16/2022).
- [75] Notion. *Notion*.
URL: <https://web.archive.org/web/20220216084535/https://www.notion.so/>
(visited on 02/16/2022).
- [76] Martin Odersky. *Scala Programming Language*.
URL: <https://web.archive.org/web/20220216084847/https://www.scala-lang.org/>
(visited on 02/16/2022).
- [77] Max Rehkopf and Atlassian. *Kanban vs. scrum: which agile are you?*
URL: <https://web.archive.org/web/20220202091611/https://www.atlassian.com/agile/kanban/kanban-vs-scrum>
(visited on 02/02/2022).
- [78] Slick. *Slick*.
URL: <https://web.archive.org/web/20220216084943/https://scala-slick.org/>
(visited on 02/16/2022).
- [79] SortableJS. *SortableJS*.
URL: <https://web.archive.org/web/20220216085018/https://sortablejs.github.io/Sortable/>
(visited on 02/16/2022).
- [80] Jeff Sutherland and Ken Schwaber. *Scrum Guide*. 2010.
URL: <https://web.archive.org/web/20220216084919/https://scrumguides.org/scrum-guide.html>
(visited on 02/16/2022).
- [81] R Core Team. *R: A Language and Environment for Statistical Computing*.
URL: <https://web.archive.org/web/20220216084751/https://www.r-project.org/>
(visited on 02/16/2022).
- [82] VentureBeat. *Trello reaches 50 million users, introduces new automation and template features*. 2019.
URL: <https://web.archive.org/web/20210627145704/https://venturebeat.com/2019/10/30/trello-reaches-50-million-users-introduces-new-automation-and-template-features/>
(visited on 06/27/2021).
- [83] Vercel. *Next.js*.
URL: <https://web.archive.org/web/20220216084503/https://nextjs.org/>
(visited on 02/16/2022).
- [84] Jean-Baptiste Waldner and Jbarta. *Kanban_principles.svg*.
URL: https://web.archive.org/web/20220216085425/https://commons.wikimedia.org/wiki/File:Kanban_principles.svg
(visited on 02/16/2022).

- [85] Wikipedia. *Asana (software)*.
URL: https://web.archive.org/web/20220216085235/https://en.wikipedia.org/wiki/Asana_%28software%29
(visited on 02/16/2022).
- [86] Wikipedia. *GitHub*.
URL: <https://web.archive.org/web/20220216085322/https://de.wikipedia.org/wiki/GitHub>
(visited on 02/16/2022).
- [87] Wikipedia. *Markdown*.
URL: <https://web.archive.org/web/20220216085533/https://en.wikipedia.org/wiki/Markdown>
(visited on 02/16/2022).
- [88] Wikipedia. *Trello*.
URL: <https://web.archive.org/web/20220216085641/https://en.wikipedia.org/wiki/Trello>
(visited on 02/16/2022).