FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

TEMPO: A Framework for Team Composition and Management in Project-Based Organizations

Dora F. Dzvonyar

FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Forschungs- und Lehreinheit 1
Angewandte Softwaretechnik

TEMPO: A Framework for Team Composition and Management in Project-Based Organizations

Dora F. Dzvonyar

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Florian Matthes
Prüfer der Dissertation:	1. Prof. Dr. Bernd Brügge
	2. Prof. Dr. Marco Kuhrmann

Die Dissertation wurde am 08.05.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 13.07.2019 angenommen.

Abstract

Rapid technological progress and dynamic customer needs force companies to move away from strict hierarchies and clearly divided functional departments toward project-based organizations. This organizational model introduces complexity in terms of team composition, which is recognized as a defining factor of the team's performance and its ability to react to change. A project-based organization does not have long-standing teams in which people work together for long periods of time: Shorter product cycles and inter-departmental teams pose challenges with respect to communication, personal relationships, and knowledge retention. The staffing process for these teams involves many different and possibly conflicting factors such as skills, culture, or personality. This can quickly become complex, especially with larger teams and an increasing number of projects in the organization.

The goal of this dissertation is to create a framework for project team composition to algorithmically support the staffing process, thus increasing its manageability. The framework leverages the capability of algorithms to take many factors into account and efficiently compute a solution which can then be modified with the skills and intuition of an experienced project leader.

In order to achieve this goal, we created TEMPO, a framework for **TE**am co**M**position in **P**roject-based **O**rganizations. TEMPO supports the project leader in the parts of the process that are most challenging and tedious for humans to master. The framework can be tailored to existing team composition processes in project-based organizations and allows the usage of different staffing algorithms.

To evaluate the framework, we developed TEASE, a reference implementation of TEMPO's team initialization workflow. We then applied TEASE in a multi-project course for mobile application development with clients from industry in a 3-year-long embedded case study. We compared the algorithmically supported team initialization process with the manual process in 33 projects to examine TEASE's impact with respect to process duration, objective value function, and number of broken constraints. We found that TEASE shortens the team initialization process by 61% while finding assignments of comparable or better quality (less broken constraints and better objective value function) than the manual process. In addition, TEASE was used for the team initialization in three instances of the project course to iteratively improve the tool by reducing the number of process gaps. Finally, an observational study was conducted with eight project leaders. The results show that the tool can be used to compose teams using pre-defined criteria with minimal training.

Zusammenfassung

Um mit dem zunehmend schneller werdenden technologischen Fortschritt mitzuhalten, bewegen sich immer mehr Unternehmen weg von strikten Hierarchien und klar getrennten Abteilungen hin zu projektbasierten Organisationen. Diese Organisationsstruktur bringt Komplexität in der Teamkomposition mit sich, denn es gibt keine langjährig existierenden Teams mehr, deren Mitglieder über eine lange Zeit zusammenarbeiten: Kurze Produktzyklen und abteilungsübergreifende Teams bedeuten Herausforderungen in Sachen Kommunikation, zwischenmenschlicher Beziehungen und Wissenserhalt. In der Zusammenstellung solcher Teams müssen viele verschiedene, womöglich miteinander in Konflikt stehende, Faktoren einbezogen werden, beispielsweise Kompetenzen, kulturelle oder Persönlichkeitsunterschiede. Der Prozess wird schnell komplex, vor allem wenn es um große Teams oder Organisationen mit vielen Projekten geht.
Das Ziel dieser Dissertation ist die Erschaffung eines Frameworks für die Zusammenstellung von Projektteams, das den Prozess algorithmisch unterstützt und so seine Handhabbarkeit steigert. Das Framework soll Algorithmen nutzen, die viele Faktoren gleichzeitig bei der Lösungsfindung einbeziehen können, um einen Vorschlag für die Teamzusammenstellung zu machen, der dann von einem erfahrenen Projektleiter angepasst werden kann.
Um dieses Ziel zu erreichen, haben wir TEMPO entworfen, ein Framework zur Teamkomposition in projektbasierten Organisationen. TEMPO unterstützt den Projektleiter in den Teilen des Prozesses, die für den Menschen am mühsamsten sind. Das Framework kann auf die existenten Teamzusammenstellungsprozesse in projektbasierten Organisationen zugeschnitten werden und erlaubt die Nutzung mehrerer Lösungsalgorithmen.
Um das Framework zu evaluieren, wurde TEASE entwickelt, eine Referenzimplementierung von TEMPO's Teaminitialisierungsworkflow. TEASE wurde im Rahmen einer dreijährigen Fallstudie in einem Multiprojektkurs mit Industriepartnern angewandt. Der algorithmisch unterstützte Teaminitialisierungsprozess wurde mit dem manuellen Prozess in 33 Projekten verglichen, um den Einfluss von TEASE auf die Prozessdauer, den Wert der Nutzenfunktion und die Anzahl der nicht eingehaltenen Bedingungen zu untersuchen. Die Ergebnisse zeigen, dass TEASE die Prozessdauer um 61% verkürzt und gleichzeitig Teams mit gleicher oder besserer Qualität findet, als dies mit dem manuellen Prozess der Fall war. Anschließend wurde TEASE zur Teaminitialisierung in drei Instanzen des Projektkurses genutzt, um das Tool iterativ zu verbessern. Schlussendlich führten wir eine Beobachtungsstudie mit acht Projektleitern durch und fanden heraus, dass das Tool mit minimaler Einweisung zur Zusammenstellung von Teams anhand vordefinierter Kriterien verwendet werden kann.

Acknowledments

No dissertation can (or should) be accomplished alone, and I am truly grateful for the support I have received along the way. First and foremost, I would like to thank my advisor, Bernd Brügge, who started supporting me already as a student, recognized my passion for applied teaching and offered me a position. Thanks for the countless discussions and rounds of refinement which made this thesis what it is. I also want to express my gratitude to my second advisor, Marco Kuhrmann, whose detailed input and guidance were incredibly useful both for the methodology of this thesis as well as for future academic work.

The past years have been some of the best so far, and this is largely owing to the colleagues I am lucky to work with. I would like to recognize in particular Dominic Henze, Paul Schmiedmayer, and Lukas Alperowitz, who have made and are making the management of our project course a breeze while never running out of crazy new ideas to keep things interesting. I am grateful for your reliability, moral support, and sense of humor. I would also like to thank Stephan Krusche for pushing me to join the Chair for Applied Software Engineering in the first place, as well as Jan Ole Johanßen, Andreas Seitz, Constantin Scheuermann, and all my other amazing colleagues for their input along the way. Of course, none of our work would be possible without the people behind the scenes: Thanks to Monika Markl and Helma Schneider for their continuous organizational support, as well as to our hard-working student helpers and system administrators, in particular Matthias Linhuber, Florian Angermeir, and Florian Bodlée. Moreover, I am grateful to the students and practitioners who agreed to participate in my research, especially Malte Bucksch, Eriks Gopaks, and Quirin Schweigert who contributed through research or development effort.

Looking beyond the Chair for Applied Software Engineering, I am incredibly thankful to my friends, sister, and parents for the continuing support and occasional poke about the progress of the thesis, which was admittedly not always met with a patient explanation from my side. I appreciate you bearing with me, accepting that academic work is often a nonlinear process and never making me feel like I should ‘get a real job’. I am also grateful to my ‘second family’, the TEDxTUM team, through which I got to meet some of the most remarkable and driven people I have the privilege of knowing. The passion, dedication and playfulness with which we operate have kept me going even through the more difficult times. And last but definitely not least, member of both families and partner in crime: A huge thanks to Julian Dlugosch for being a caring, encouraging, and inspiring partner and making me a better person overall.

Contents

Conventions

We use American English, except for literal quotations. Inline citations are accentuated by “double quotes”. Changes to direct citations are marked [within brackets].
We capitalize technical terms (e.g. Analysis Object Model) and components of the framework (e.g. Person Pool, Instructor Rating), as well as elements of models and figures except when they are only paraphrased in the text (e.g. `initializing a team` vs. `Team Initialization` use case).
A `typewriter font` is used to highlight classes, objects and activities in UML models as well as elements of figures in their direct description. We use *italic font* to mark technical terms the first time they appear in the text, for highlighting a term when we describe multiple terms after each other, for metric and variable names (e.g. *total duration of team initialization*), as well as for TEMPO workflows (e.g. *Team Initialization* workflow), levels of Instructor Ratings (e.g. *Novice*) and constraints (e.g. *development devices* constraint).
This document contains colored figures and should therefore be printed in color.

Chapter 1

Introduction

> People matter in building software. [...] Tools matter. Techniques also matter. Process, yet again, matters. But head and shoulders above all those other things that matter are people.
>
> Robert Glass

In order to keep up with fast-evolving demands and markets, rapid technological progress, and dynamic customer needs, companies are moving away from the traditional line or divisional organization with strict hierarchies and clearly divided functional departments. Some use a matrix organization based on *project teams* that group members from the different lines together into a unit that is responsible for a project from beginning to end [CL04; HG92; KW97; THCG04]. These project teams are said to "provide the needed depth and breadth of skill and experience" [DeF02] especially when designing or producing complex products and services. Some companies even adapt their organizational structure to become *Project-based Organizations* "in which the project is the primary unit for production organisation, innovation, and competition" and there is no need for the formal departments which made up the lines of the traditional organizational structure [Hob00].

These new organizational models introduce complexity and change in terms of human resources, since they cannot rely on long-standing, steady teams in which people work together for years: Shorter product cycles and inter-departmental teams pose challenges around communication, personal relationships, knowledge retention and many more areas [DeF02; KW97]. Hierarchies and responsibilities are blurred and

need to be re-defined in the employees' minds [HG92], team members who do not know each other cannot be expected to have the same background and work habits [Ree99], and projects with distributed or part-time members add complexity in collaboration [SLPK15; THCG04]. The fact that people go through more jobs during their career than in the past [Bur18; RS14] because they increasingly strive for a motivating, purposeful job that fits their interests, skills, and lifestyle [Pee06; Ste16] introduces even more change. Addressing these challenges is especially important in a creative field like software engineering which requires a variety of skilled individuals to work together. Some even argue that "[s]ooner or later, major issues relevant to software engineering boil down to the people involved with software production" [CA10].

The composition of a project team, i.e., the combination of people on a team and their individual characteristics, is recognized as a defining factor of the team's performance [CL04; PSCB00; Ree99; SG10; WWB07]. However, team composition is a problem involving many, possibly conflicting, factors. For instance, it is not sufficient to staff each position individually with an employee who possesses the right skills to fulfil that specific role; interactions between the team members have to be taken into account [CA10; CL04; SLPK15; WWB07]. In addition, group-level skills that are not directly related to the task, but help the group to function (e.g. a collective ability to learn and take initiative, stress tolerance, and flexibility when confronted with change) also have to be present in the team for it to perform well [PSCB00]. This quickly becomes complex with a growing amount of employees and projects in the company, increasing the amount of possibilities for staffing each position [THCG04; WWB07].

Human resources management activities and processes are more and more supported by algorithms or automated altogether. An analysis by McKinsey suggests that "56 percent of typical 'hire-to-retire' tasks could be automated with current technologies and limited process changes" [BG18], leading to increased efficiency, cost savings, and a more unified experience of employees. Algorithmic automation, Artificial Intelligence and cognitive computing tools have been identified to have large potential for recruiting and talent acquisition, employee onboarding, talent development, performance measurement and reporting, as well as with repetitive HR operations tasks such as creating and managing documents, tracking time and availability, or generating reports [Hum17; IBM16]. Considering their capability for making sense of large amounts of structured and even unstructured data, these approaches have great potential to be used in team composition as well. However, algorithms should not be used as a "cookie-cutter solution" capable of replacing human judgement [Hum17]: One should find the 'sweet spot' at which the benefits of an algorithmic approach are leveraged to support the human project manager with the right knowledge at the right time to make better decisions in complex and uncertain situations.

The main goal of this dissertation is to create a framework for project team composition and management that algorithmically supports the project leader's decisions. The following sections describe the problem and the research approach in more detail.

1.1 Problem Statement

A team's composition is regarded as an important factor influencing its performance [CL04; PSCB00; Ree99; SG10; WWB07] and there is a variety of academic work on how e.g. cultural fit, demographics, compatibility of work habits, personality, motivation, and soft skills of team members affect their work [AJ04; DM01; Fra+11; KYR06; YKM08]. However, these studies only analyze a few factors at a time: Taking into account a larger number of variables and their possible interdependencies increases the complexity of the problem until it becomes impossible to handle for humans, especially in organizations with a large pool of employees [SG10]. Several algorithmic approaches to team composition have been presented to tackle this challenge [BBW08; CCFC12; FA05; THCG04]. They vary in terms of the algorithms used, the supported input parameters and their tolerance toward incomplete or fuzzy information. Moreover, while these approaches produce one or multiple mathematically optimal solutions to the assignment problem, they fail to take into account the subjective experience of the project leader, since human intuition cannot be modeled to be fully understood by an algorithm.

Considering these drawbacks, it is not surprising that the availability and adoption of tools for team composition in industry are subpar [SG10]. An interview study with practitioners revealed that while especially larger organizations have defined company-wide processes to compose teams and tool support for e.g. searching for employees with fitting qualifications, they prefer their personal networks and informal routes to find suitable candidates [DB18b]. Tools are used if informal knowledge is not available, for instance when staffing a project with employees from an offshore development center. Current tools on the market include automation software for recruitment and onboarding,[1] for regular HR tasks such as vacation requests or time sheets,[2] or tools for managing work in running teams.[3] For team composition, however, little tool support exists beyond locating employees with specific skills. Although the majority of interviewees sees the potential of using algorithmic support or even expects it to lead to better teams, they do not rely on currently available tools because they find them inadequate: The failure to document experience in a sufficient accuracy

[1] e.g. `https://www.laserfiche.com/solutions/human-resources/`
[2] e.g. `https://kissflow.com/`, `https://wperp.com/hr/`
[3] e.g. `https://hubplanner.com/`, `https://resourceguruapp.com/`, `https://www.float.com/`

and granularity as well as the missing representation of soft factors like interests or motivation were mentioned as their major shortcomings [DB18b].
These findings are in line with other existing research: According to a 2017 survey of Human Resources Professionals Organization members [Hum17], 84% believe that algorithmic approaches, notably Artificial Intelligence, are a useful tool for HR professionals, but over two thirds don't feel that their workplace is prepared to use people analytics and AI to address HR challenges. The 2016 IBM Institute for Business Value Cognitive Computing Study [IBM16] found that executives also recognize the value of algorithmic solutions, but are unsure how to proceed. While there are some case studies mentioning the use of algorithms in recommending job opportunities to individual employees in large organizations like Salesforce or General Electric [Hum17] or filling singular open positions with fitting candidates [IBM16], there is little to no work reported about a formalized, replicable use of algorithmic decision support in composing teams.
In summary, the following problems were identified:

Problem 1 The manual composition of project teams is challenging especially in large organizations due to the large number of variables that need to be considered.

Problem 2 Purely algorithmic team composition does not take into account the valuable experience of managers.

Problem 3 The current tool support for composing teams is inadequate and thus only relied upon when personal knowledge is not available.

In the next section, we describe our research approach to address these problems.

1.2 Research Approach

The goal of this dissertation is to create TEMPO, a framework for **TE**am co**M**position in **P**roject-based **O**rganizations. TEMPO's aim is to make the team composition process of project-based organizations more manageable by algorithmically supporting the staffing of project teams. In order to ensure that TEMPO addresses current challenges, we base the process on an analysis of the state of the art and then incrementally develop a solution and validate it in the usage context. In the following, we explore approaches with similar aims and develop the research approach for this dissertation.
O'Leary and Richardson [OR12] describe their strategy for developing a process reference model using a multi-method research design. Their approach is evolutionary

and based on data from both industry and academia: The development of the model takes place in stages, each of which validates the current version of the model and implements the identified improvements. The first version is based on expert opinions and literature research to achieve "an analysis of current domain knowledge through the construction of a frame of reference" [OR12], and subsequent versions are validated using different empirical methods. For instance, they use the first stage of their solution to conduct an industrial case study, and the results of this study then serve to develop the second version, which is evaluated using comparative analysis to complement the initial qualitative feedback with quantitative results. They describe these stages as "a continuum in which the focus of the research [...] is continually adjusted based on the results of the previous stage" [OR12].

The *Technology Transfer Model* (TTM), shown in Figure 1.1, is a methodology for the "successful transfer of knowledge and technology from research to practice" [GGLW06]. The first step in this approach is to identify real problems in industry through observation and analysis, based on which possible research topics and problem statements are formulated. The subsequent development of a candidate solution takes place in close collaboration with practitioners who can verify requirements along the way. The candidate solution should be validated in a 'lab', a controlled academic setting, to identify flaws before the solution is piloted. Following this, the TTM suggests a static validation, for instance through presentations or seminars in industry, to be able to further improve the solution based on feedback. Before the solution is released, dynamic validation takes place with a pilot version [GGLW06; Woh+12].

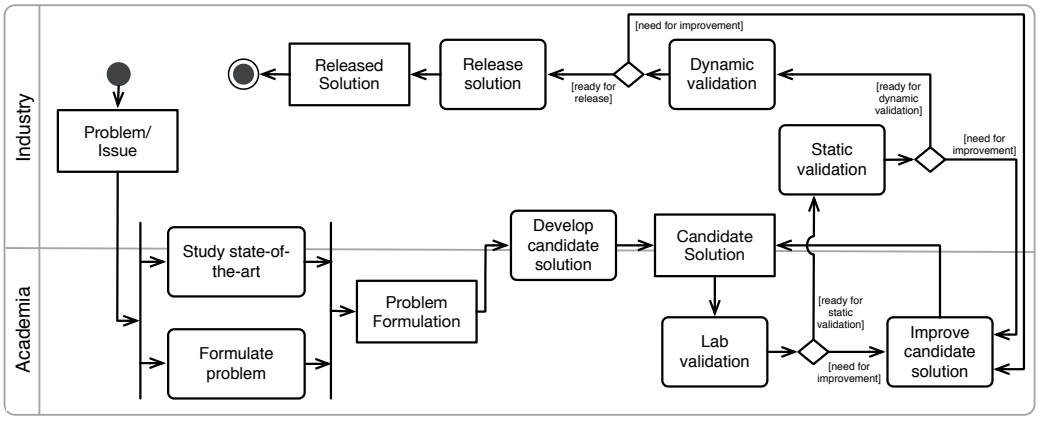


Figure 1.1: Technology Transfer Model, adapted from [GGLW06]

The Unified Software Development Process [JBR99] is a lifecycle model that proposes an iterative and incremental approach to software development. It consists of an *Engineering Stage* with its two phases *Inception* and *Elaboration* and a *Production Stage* with its two phases *Construction* and *Transition*. Each of the four phases can

consist of multiple iterations which produce an internal artifact. Before the decision to transition to the next phase takes place, a formal, stakeholder-approved artifact is produced. Workflows such as *Management*, *Requirements*, *Design*, or *Implementation* are carried out in different intensities throughout the lifecycle, depending on the necessity of the activity in the current phase and iteration [JBR99].
The research approach of this dissertation contains elements from these three approaches: It instantiates the Technology Transfer Model and extends it as shown in Figure 1.2.

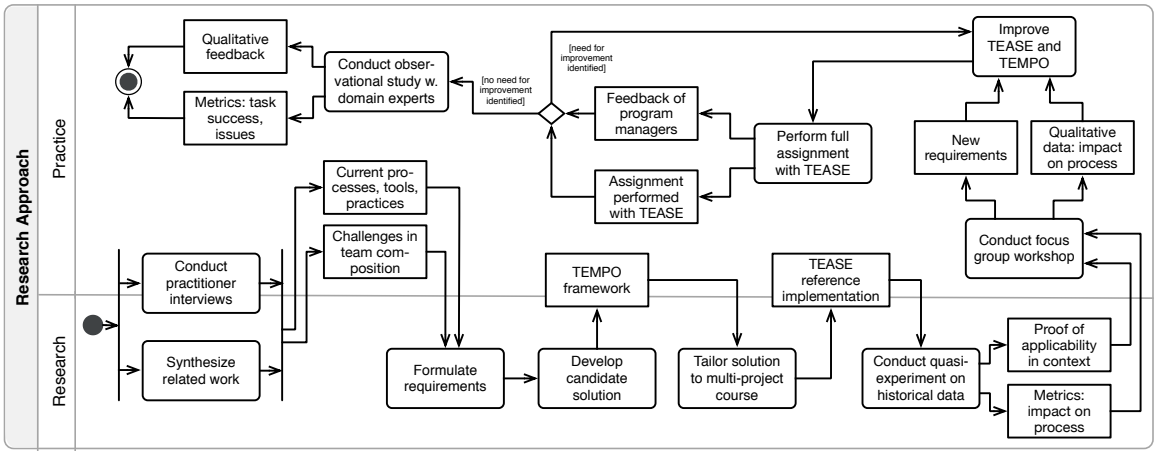


Figure 1.2: Overview of research approach

The motivation for this research is based on industry needs, elicited through research in practice in the form of an interview study, in combination with the synthesis of related work as recommended by both the TTM and O'Leary and Richardson [OR12]. The aim at this stage is to understand current processes, tools, and practices, as well as challenges in team composition in project-based organizations. Based on the identified needs, we analyze the problem domain and formulate the requirements for a candidate solution.
We develop the TEMPO framework with its workflows *Project Preparation*, *Team Initialization*, *Team Composition Monitoring* and *Employee Information*. The scope of TEMPO are organizations that use project teams, allowing customization and tailoring to the individual organization's processes. Moreover, the framework supports refinement of the criteria for composing future project teams, allowing the organization to adapt its approach as it matures and evolves.
Since the development of TEMPO is iterative and incremental, it can also be based on the Unified Software Development Process. As this approach is primarily meant to cover the development of software projects, however, it assumes that the construction phase will consume the most time and effort [JBR99], which is not the case with the development of TEMPO. Instead, there is a bigger focus on empirically validating the

impact of the framework on the usage context. In order to evaluate the applicability of TEMPO, we instantiate it in the iPraktikum, a multi-project course for mobile application development with 80-100 developers and 10-12 projects running in parallel [KABW14]. The course uses flat staffing of balanced software engineering teams in a purely project-based organization [DAHB18]. We design and develop TEASE, the **TE**am **A**llocator for **S**oftware **E**ngineering courses. TEASE is a reference implementation for TEMPO's *Project Preparation* and *Team Initialization* workflows in this setting.

The evaluation of the solution takes place in the context of this large multi-project course. While the TTM envisions a validation directly in industry, this is not feasible within the scope of this research because of the lack of access to employee data and a reluctance to adapt the existing process due to security concerns and cultural legacy in the studied organizations. Especially large organizations are known for their persistence and resistance to change and are more likely to adopt new tools and processes after they have been validated elsewhere [PRW13]. An interview study with practitioners revealed that while the majority of interviewees is convinced of the usefulness of such a tool, their organizations would currently not allow its adoption due to privacy concerns, requiring the tool to go through a lengthy process of validation which is not in the scope of this dissertation [DB18b]. Hence, we opt for the application of the framework in a setting which is close to industry and at the same time open to be empirically studied and where it is under the control of the researchers to make adaptations based on the findings.

We follow the validation steps of the TTM and extend them based on the recommendation by O'Leary and Richardson [OR12] to validate the solution in multiple stages using different research methods. The evaluation of TEASE is done through an embedded case study [RHRR12] in which we investigate multiple units of analysis within the same case (the team initialization process) and in the same context (the iPraktikum). Since we want to iteratively improve TEMPO and TEASE, we use elements of action research, which is closely related to the case study, but "focused on and involved in the change process" [RH09a].

The Technology Transfer Model suggests to validate the candidate solution in a controlled setting before transitioning to the target environment. We therefore evaluate TEASE's applicability to the context of the project-based course by comparing it to the previous manual process. In a quasi-experiment, the program managers of the course perform the team initialization using historical participant input data. Based on the Goal-Question-Metric Model [SBCR02], we measure the metrics *total duration of team initialization*, *duration of each process step*, *objective function value of assignment* and *number of broken constraints* to determine the differences between manual and

algorithmically supported team initialization.
As static validation, we conduct a focus group workshop with the program managers. This workshop has both an explanatory and an exploratory purpose: We want to to enrich the quantitative data from the lab validation with qualitative feedback and elicit new requirements for TEASE. The research questions investigate the program managers' experience performing the team initialization with the tool as well as requirements that need to be fulfilled before the tool can be released.
Following the static validation, we improve TEASE and transition to the first dynamic validation step by performing the full team initialization using the tool in three consecutive instances of the project course. We use action research as opposed to a purely observational case study: After each initialization, we collect feedback from the program managers and adapt TEASE accordingly. This formative approach allows us to iteratively examine the usefulness of the solution while it is being developed instead of assessing it after it has been released. In each iteration, we look for gaps between the current functionality of TEASE and the manual team initialization process, as well as newly emerging requirements that go beyond the original process.
After three iterations of TEASE in the context of the multi-project course, we conduct an additional dynamic validation step in the form of an observational study with project leaders. The aim is to evaluate the usability of TEASE with subjects that have led teams in this context for multiple years and are thus experts in the application domain, but have not yet worked with the tool. We collect qualitative feedback using the think-aloud protocol, quantitatively measure the success rate of the subjects as well as how many and what kind of issues they experienced, and follow up with a questionnaire asking about their experience. We evaluate if new users can successfully perform the team initialization with minimal prior training, examine the issues they faced as well as how challenging they found the tasks, and measure their satisfaction through perceived usefulness, perceived ease-of-use and open questionnaire responses.
The result of these evaluation steps is feedback on the applicability of TEMPO and TEASE to this multi-project context, both in the form of qualitative data from the program managers and project leaders, as well as a quantitative measurement of the assignment quality and duration. The release and evaluation of TEMPO in industry remain future work.

1.3 Contributions and Scope

This section describes the contributions of this dissertation, defines its scope, and lists previously published material.

Contributions

The first contribution of this dissertation is an analysis of team composition in practice. We assess algorithmic approaches to team composition by synthesizing related work in Section 2.2.4. Moreover, we analyze the results of an empirical interview study with practitioners in project-based organizations in the software industry with regard to their processes and roles for team composition Section 2.3.
Based on the identified challenges, we develop the TEMPO framework in Chapter 3. TEMPO algorithmically supports the project leader when composing project teams and offers the workflows *Project Preparation*, *Team Initialization*, *Team Composition Monitoring*, and *Employee Information*. When composing a team, the project leader can set constraints and objectives and gets algorithmically optimal suggestions from TEMPO. These suggestions can then be adapted based on human knowledge and experience with the people involved, and TEMPO communicates the impact of the changes on the team's composition to the project leader. The framework is the main contribution of this dissertation.
In order to evaluate TEMPO, we instantiate it in the context of a large multi-project capstone course in Chapter 4. We develop TEASE, a reference implementation that is tailored to the staffing process used in the course. TEASE is then evaluated in an embedded case study in Chapter 5 using the steps lab validation, static validation and dynamic validation suggested by the Technology Transfer Model [GGLW06] and described in Section 1.2. The instantiation of TEMPO and the empirical evaluation of TEASE conclude the contributions of this thesis.

Scope

Team composition is a very broad topic that can be studied from many sides in different domains. Therefore, the scope of this dissertation has to be narrowed down and defined.
TEMPO is a framework intended for pure project-based organizations, project team organizations and matrix organizations. Each of these organizational models is described in detail in Section 2.1. For the scope of this research, we focus on organizations in the software industry. TEMPO offers workflows from the initialization of a new project team until the termination of a project.

TEASE is tailored to the team initialization process of the multi-project capstone course, the focus of which is the flat staffing of 10-12 teams from a pool of 80-100 developers without specialized roles [KABW14]. Since this is an educational setting, it is important to balance experience of developers in teams while taking criteria such as project priority, team size, gender, or development and test devices into account [DAHB18]. Moreover, it is assumed that accurate and up-to-date data is available about all participants. Although TEASE likely supports larger pools as well as gradual staffing of teams, so far its evaluation is limited to this setting.
Both TEMPO and TEASE are aimed at increasing the manageability of the team composition process. This is achieved by algorithmically supporting the staffing of project teams based on pre-defined criteria, providing the necessary data on projects, employees and teams at the time and in the format in which it is needed, and visualizing the effect of changes in the team's composition on the selected criteria. We do not claim that this increases the quality or performance of the resulting project teams, since this depends on a variety of other factors and the effect of team composition on team quality cannot be considered in isolation. The criteria suggested in TEMPO and instantiated in TEASE are based on a review of the literature regarding factors that benefit team performance, an analysis of what practitioners consider to be important criteria, as well as our experience in composing teams for project courses since 2008. TEMPO also allows for a refinement of the criteria based on the outcome of previous projects, thus allowing for a learning process in the organization.

Previously Published Material

Parts of the contributions of this dissertation have been previously published:

- [DB18b] contains the first results of an interview study with practitioners in project-based organizations in the software industry. We described and discussed observations regarding the state of the art of team composition processes and tools in practice. This dissertation builds on some of the results reported there: Based on the observations, we classify roles and types of team composition processes in Section 2.3 and also use the results to identify visionary scenarios and requirements for TEMPO in Chapter 3.

- In [DAHB18] we published an overview of the manual team initialization process and its underlying criteria in the multi-project course iPraktikum. In this dissertation, we build on this process by designing a framework that can be tailored to this process. We provide a reference implementation that instantiates the framework in this context. During the dynamic validation in Section 5.3, we

go beyond the original, manual process as we iteratively improve the framework and reference implementation.

- We previously published some of the results of the first validation of TEASE in [DHAB18]. In this dissertation, we perform a lab validation of TEASE with the same approach using data of 22 additional projects in Section 5.1 and conduct three more evaluation steps in Sections 5.2 to 5.4.
- The multi-project course iPraktikum serves as the context for the instantiation of TEMPO. Previous publications exist that do not have team composition as their focus, but describe other workflows or educational approaches within the same setting, e.g. [ADB16; AJDB17; DKA14; DB18a; KDXB18].

1.4 Dissertation Structure

The remainder of this dissertation is structured as follows.

Chapter 2 establishes the foundations by presenting relevant theory as well as related work. After defining necessary terms, we introduce traditional and modern organizational structures, summarize theory around teams and team composition, and describe the current state of team composition in practice.

Chapter 3 presents TEMPO: We elicit the visionary scenarios as well as functional and non-functional requirements for the framework based on the foundations presented in the previous chapter. We further describe the framework in terms of use cases, models, and algorithmic design, and discuss prerequisites and limitations of its use.

Chapter 4 presents the instantiation of TEMPO in the form of TEASE: We tailor the framework to the team initialization process we use in the multi-project course. We describe the context of the reference implementation as well as the current manual process, show the instantiated workflows and explain technological and implementation details of the tool.

Chapter 5 describes the evaluation of TEASE in the multi-project course iPraktikum. We perform the steps *lab validation*, followed by the *static validation* and two subsequent *dynamic validations* in the target environment. We present the research design, data collection and results and then discuss the findings for each step. Finally, we reflect on limitations and threats to validity of all validation steps.

Chapter 6 summarizes the contributions of this dissertation and gives an outlook on future work and directions for further research.

Chapter 2

Background & Related Work

> The best executive is the one who has sense enough to pick good men to do what he wants done, and self-restraint to keep from meddling with them while they do it.
>
> Theodore Roosevelt

This chapter establishes the foundations of this dissertation by presenting relevant theory, related work and own research carried out as basis for TEMPO. Section 2.1 introduces traditional and modern organizational structures and current trends. Section 2.2 summarizes theory around teams and team composition, including both personal and team-level factors as well as group development models and approaches to algorithmic team composition. Finally, Section 2.3 describes the current state of team composition in practice: We analyze interviews with practitioners with regard to project states, roles, and process types in team composition in project-based organizations.
The findings of this chapter serve as the basis for the problem formulation and requirements of TEMPO as described in the research approach in Section 1.2.

2.1 Organizational Structure

The organizational structure defines how activities and decisions are coordinated and carried out within an organization [RJ15]. Jones [Jon10] defines two dimensions of differentiation in an organizational structure: *vertical differentiation* refers to the distribution of decision power and authority between the levels of the structure, while *horizontal differentiation* describes the grouping into roles and units of the organization according to tasks and responsibilities. The former is also referred to as *chain of command* or *decision rules*, while the latter is also called *departmentalization* [Bal06; RJ15].

In terms of vertical differentiation, a *mechanistic structure* is defined by centralized decision-making, clearly specified and formal flow of information through the hierarchy and defined, non-overlapping tasks for each role [Jon10]. On the opposite end of the spectrum, an *organic structure* is "flat, uses cross-hierarchical and cross-functional teams, has low formalization, possesses a comprehensive information network, and relies on participative decision making" [RJ15]. While mechanistic structures work efficiently with clearly defined and static processes and tasks, organic structures are designed to also perform well in an uncertain organizational environment. Laloux [Lal14] presents a different terminology and calls organizations with a mechanistic structure *traditional conformist organizations*, while those with an organic structure are *postmodern pluralistic organizations*.

Bruegge and Dutoit [BKW12] define three major types of interaction that determine the organizational structure: *Reporting* (exchanging status information, e.g. between developers or toward management), *Decision* (propagating decisions or resolving issues), and *Communication* (exchanging all other types of information). According to their definition, *hierarchical organizations* are characterized by unidirectional reporting and decision structures: Decision information only flows from the root of the organization to its leaves and status information is generated at the leaves and reported to the root [BKW12].

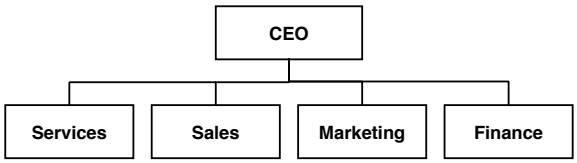


Figure 2.1: Functional organizational structure

When it comes to horizontal differentiation, a *functional structure* "groups people together on the basis of their common expertise and experience or because they use the same resources" [Jon10]. This results in functional departments such as `Sales`

or `Marketing`, as depicted in Figure 2.1. A *divisional structure* groups functions according to the demands that they serve, which can be those of a product, a market (e.g. commercial vs. government), an industry or a geographic area [Jon10]. In an organization providing software and services, a grouping according to industries could look as shown in Figure 2.2, with individual divisions for `Automotive`, `Healthcare` and `Banking` products. Each of these has their own departments that are focused on the individual needs of their specific industry.

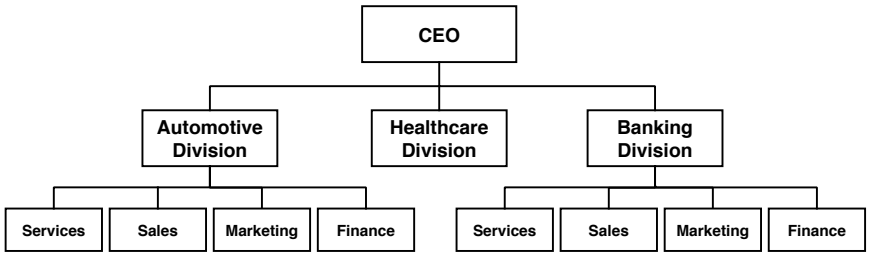


Figure 2.2: Divisional organizational structure

The *matrix structure*, shown in Figure 2.3, groups functions according to two dimensions: vertically by function and horizontally by e.g. product or project. The result are cross-functional `Product Teams` in which each member has two superiors, their product manager and their functional manager. This type of organization is also called *secondary organizational structure*, as opposed to the hierarchical, permanent *primary organizational structure* [BK13]. While the matrix organization is beneficial to facilitate coordination within the functional departments of the organization, the possible role ambiguities and conflicting demands can lead to challenges for employees [RJ15]. Moreover, the matrix organization "lacks a control structure that allows employees to develop stable expectations of each other" [Jon10].

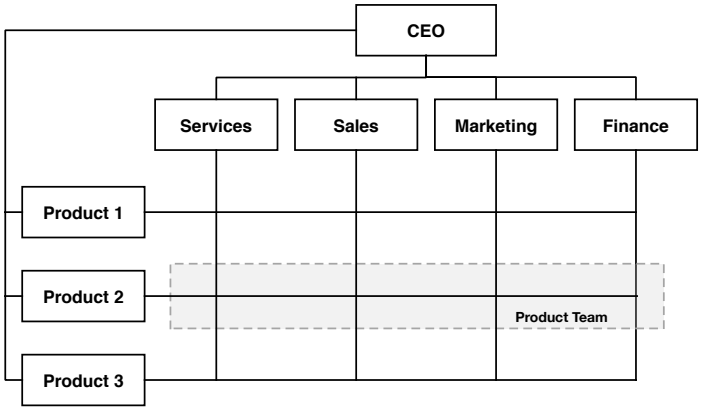


Figure 2.3: Matrix organizational structure

A *project team structure* or *product team structure* is also based on temporary cross-functional teams which are composed of specialists of the different functional departments. In this model, a team is a self-contained division responsible for the project's or product's success, and employees no longer owe allegiance to their functional superior, avoiding the 'double boss' challenge of the matrix structure [Hob00; Jon10]. Today's software and technology consulting organizations typically use this organizational structure.
A more extreme form of the project team structure is the *project-based organization (PBO)*, in which "the project is the primary unit for production organisation, innovation, and competition" [Hob00]. In this model, depicted in Figure 2.4, a team is created as an independent sub-unit for the duration of the project and there is no formal coordination across project lines [DeF02]. The team is typically self-organizing and the project leader has high decision capability. A project-based organization is typically used when developing high-value, complex products and services and has been adopted by various companies in e.g. the technology, manufacturing, transportation or film industries [Hob00]. This organizational structure is suitable for operating under uncertainty, in a fast-changing environment where cross-functional expertise is needed, or where a strong customer focus is necessary. However, since the project team is dissolved after the project and the employees are scattered over other projects within and outside the organization, knowledge retention and the coordination of capabilities and resources across the organization are challenging [DeF02].

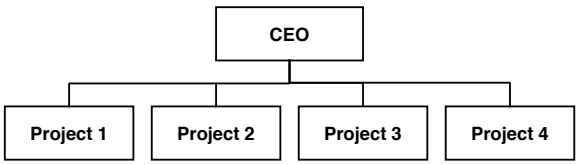


Figure 2.4: Project-based organizational structure

Some less traditional organizational paradigms have emerged in the last two decades, many of which are not defined structures but a set of practices or values to create organizations. A *community web*, typically known from open source software, is based on a community of individuals dedicated to work towards a shared goal, each contributing based on intrinsic motivation. The structure can appear chaotic from the outside, but there is an underlying order which evolves based on the community's needs, with a set of roles that are taken on voluntarily and a reputation-based control system among peers [DeF02]. The *boundaryless organization* is focused on eliminating vertical and horizontal boundaries within the company, as well as blur the boundaries toward other organizations [RJ15]. A *chaord* or *chaordic organization* balances aspects of chaos and order to be "infinitely malleable yet extremely durable" [Hoc95]. It is

self-organizing and built with the distribution of power, ownership and governance in mind and is known as the model used at VISA. Finally, a *Holacracy* is described not as a model but a practice and replaces the traditional organizational structure with circles that are self-organizing but not self-directed, meaning that they depend on and coordinate with with other circles. Within the circle, which can be e.g. a single team or a functional grouping of other circles, roles are defined as needed to replace job descriptions, and a person can hold multiple roles. Principles such as dynamic steering, which suggest that decisions should be taken based on the currently optimal solution according to present information and refined continuously, are important components of the practice and are aimed at making organizations more flexible and better at utilizing knowledge across the organization [Rob07].

2.2 Teams and Team Composition

This section presents relevant theory and related work around teams and team composition with a focus on engineering and software development teams. As defined in Appendix A, the term *team composition* as a *noun* refers to the collection of characteristics of members of the team, while the *verb* denotes the activity of assigning one or multiple persons to a team, thus changing its composition.
Section 2.2.1 describes factors that have shown to impact teamwork, while Section 2.2.2 summarizes models for generating teams. Section 2.2.3 presents theory on how teams develop over time, and Section 2.2.4 summarizes research on algorithmic team composition.

2.2.1 Factors Impacting Teamwork

Teamwork is impacted by characteristics of the group as a whole as well as the characteristics of its individual members [CL04; Ste06; WWB07]. There is a large amount of literature on team factors, and a multitude of them have shown to have an effect on teamwork and on team performance. This section summarizes theory and related work in this area, but does not give a complete overview of all variables that can influence team performance. Instead, we select a subset of factors that have shown to be most affected by team composition.
Purna Sudhakar et al. [PFP11] classify factors that affect software development team performance based on secondary research and elicit the four categories `environmental factors`, `technical factors`, `non-technical (soft) factors`, and `organizational factors` as shown in Figure 2.5. Within their classification, the factors that are most affected by the team's composition are in the non-technical category: For instance, cultural fit of a person to the overall organization is more

important during the hiring process, and environmental factors such as budget or organizational politics can impact team performance, but cannot be fundamentally changed by adapting the team's composition [RJ15].

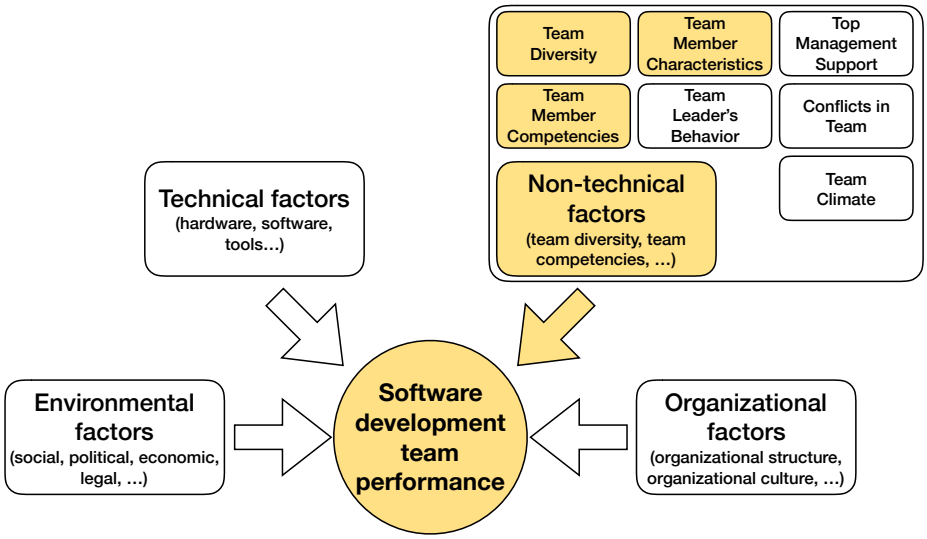


Figure 2.5: Factors affecting software team performance, adapted from [PFP11], with factors that are most affected by team composition marked in color

In this category, Purna Sudhakar et al. list the following soft factors: `team climate`, `team diversity`, `team member competencies and characteristics`, `team leader's behavior`, `top management support`, and `conflicts in team` [PFP11]. Team climate refers to the vision and level of task-orientation of a team and is therefore a factor that is more strongly affected by the processes of the team than its composition, just like the team leader's and management's behavior. While conflicts in the team and the way they are addressed do depend on the people involved, this factor is also more of a behavioral one. Therefore, the factors `team member competencies`, `team member characteristics`, and `team diversity` are most strongly affected by the team's composition, as shown in color in Figure 2.5.
In the following, we examine theory and related work for each of these three non-technical factors. Moreover, `team member motivation` is included as an additional factor, since it has been widely recognized for its impact on both individual and team performance [Bee+08; Fra+11; LB02].

Team Member Competencies & Skills

The competencies of individual members affect the team's performance by limiting the team's collective ability to perform, especially when it comes to complex tasks [RJ15; Yuk13]. Hughes and Cotterrell [HC99] identify four types of tasks and how they are affected by the skills of the people working on them: with *additive tasks*, the efforts of each participant create a summed-up result in which each individual's input counts directly, while *compensatory tasks* (e.g. code review or agile estimation) group the efforts of all participants, making it possible for more advanced members to compensate for mistakes of less experienced ones. *Disjunctive tasks* have only one answer that somebody has to come up with, making the group as good as its best member, and *conjunctive tasks* depend on everyone completing their assignment, so that the group's speed is determined by its slowest member [HC99].
Software engineering is a creative craft that needs a wide variety of skills, both for writing code as well as for other activities such as requirements elicitation, modeling, or design [Cap03; RIRJSV10]. While skills can be developed through feedback, training, or peer instruction, this factor should be taken into account when composing a team in order to ensure a necessary minimum level of skills is present in the team. Academic research has shown that the skill level and domain experience of a software team are positively correlated with product quality [BS06]. On a team level, knowledge diversity has been observed to increase task conflict in a team, which in turn is beneficial to the outcome of a task in software development [LLLL07]. In education, skill-related issues have been reported to be among the more common problems in student projects, resulting in additional workload [AP09; Bos+11; DAHB18]. In addition to technical competencies, soft skills such as commitment, eagerness to learn and communication have been shown to be regarded as important by team leaders and developers [MRF15].

Team Member Characteristics & Personality

An individual's personality determines their learning, behavioral, and decision styles and thus impacts their performance in a specific context or role [HA86]. *Personality-job fit theory* claims that satisfaction and job turnover are directly dependent on how well an employee's personality fits to their current task [RJ15].
From the variety of personality tests and inventories in the literature, the *Myers-Briggs Type Indicator* (MBTI; [MM10]) and the *Five-Factor Theory* (FFT; [MCJ99]) are among the most frequently used in studies of software engineering [Cru+11]. The MBTI is a self-reported personality inventory which is based on the four dichotomies *extraversion/introversion*, *sensing/intuition*, *thinking/feeling*, and *judging/perceiving* [MM10]. These are pairs of preferences along which individuals are characterized,

although the MBTI does not rank the preferences or indicate level of ability of a person. Nevertheless, the preferences of an individual to perform a certain task can be inferred: For instance, *sensing* types are supposedly better at going into details, while *intuitive* types prefer abstract thinking.

Capretz and Ahmed [CA10] use job descriptions for software engineering positions to create a categorization of hard and soft skills required for different roles, then mapping these to MBTI preferences. While their approach may be debatable, the study shows the variety of personalities needed for software engineering. For instance, they show that system analysis requires other preferences than programming: the former favors *extraversion* and *feeling* types for communication between stakeholders, while the latter requires more *introversion*, *sensing*, and *thinking* types for structured decomposition of the problem into code. Gorla and Lam [GL04] map MBTI types to roles in software teams and measure the effect of the project leader's and programmers' personality on perceived team productivity. Amongst other effects, they find that teams with more extroverted programmers were perceived to be more performant than those with their introverted counterparts. While perception does not always conform to objective performance, it can have an impact on team morale and motivation.

Five-Factor Theory explains how the 'Big Five' personality traits *Openness to Experience*, *Conscientiousness*, *Extraversion*, *Agreeableness*, and *Neuroticism* relate to behavior and preferences [MCJ99]. For instance, individuals that score high on *Agreeableness* are supposed to be more cooperative than those with a lower score, while *Conscientiousness* contributes to a dependable, goal-oriented work style [KW97]. In terms of tests to measure these factors, short scales such as the *Ten-Item Personality Inventory* (TIPI; [RVGFLR12]) exist and have been shown to be comparably robust to a full test. In terms of empirical studies, Salleh et al. [SMG11] showed that students who score higher in *Openness* not only perform better in pair programming tasks, but also enjoy them more. Assuming that some people generally do not enjoy working in teams as much as others, the consideration of personality has even more important implications for team composition.

Personality has also been studied on a group level using measures such as the MBTI or FFT, but the measurement of a 'team personality' is challenging, since the individual members' traits are, to some extent, lost by aggregating them to an overall score [KW97; Pee06]. Some researchers therefore use more detailed measures containing a combination of the summed up team score of each trait and the degree of variation between team members [NWC99; Pee06]. In such studies, heterogeneity of personalities has shown to be beneficial when tackling diverse tasks, but also a source of disagreement or conflict, especially when it comes to differences in scores measuring reliability and emotional stability [KW97; LLLL07; NWC99]. Taking into account the fact that

different personalities are needed at different stages of a project, or that the team as a collective needs a minimal threshold of certain traits (e.g. emotional stability) to be able to function at all [KW97], calculating a team personality profile using individual measures such as the MBTI or FFT quickly becomes complex. Several models exist that aim to help with composing well-functioning teams using more generic roles. These models are further discussed in Section 2.2.2.

Team Diversity & Cohesion

Diversity refers to the degree of variation in attributes such as culture, gender, race, age, or education across team members. It has shown to impact the performance of a team both positively and negatively, although the results of academic work in software engineering are not entirely clear. When team members strongly differ in one or more traits, this can create a perceived divide and split the group into multiple sub-units that do not communicate as well [RJ15]. A lack of cultural fit within a team has been found to be a source of demotivation, communication problems and conflict in both industry and educational studies [Bos+11; DM01; FPC01; Fra+11; LLLL07; Ojh05]. The results of experiments studying gender in software engineering teams have not yielded clear results [FSM12; Ojh05], but research has shown that fostering collaboration across genders can break down stereotypes [Har97].
Cohesion is the degree to which a group acts as a whole and its members feel a sense of commonality and belonging. Mutual trust and cooperation are especially important with highly interdependent roles and can be increased through promoting a shared vision, celebrating team rituals, performing team building activities, or using external motivators such as rewards [Yuk13]. While these are activities that happen when the team is already working together, cohesion can also be influenced by assigning people who are likely to get along well with each other to a team. A high level of cohesion can cancel out the negative effects of perceived differences [RJ15] and has shown to have a stronger effect on team effectiveness than demographic similarities [KYR06]. However, a too high level of cohesion can lead to *groupthink*: due to the high level of similarity of opinions, team members no longer adequately question their decisions [RJ15; Yuk13]. Conflict is beneficial up to a certain extent, depending on its nature: research has shown that task-centered conflict has a positive effect on software team performance, while relationship conflict affects it negatively [LLLL07].

Team Member Motivation

The motivation of individuals on the team also influences its output, since demotivated team members are less productive, take more sick-leave and are more likely to leave the organization altogether [Bee+08]. A wide variety of motivational models exists in the literature, many of which focus on leadership: for instance, *Theory X* assumes that employees are fundamentally lazy and need external forces to motivate them to perform, while *Theory Y* supposes that people want to perform well in their work and leaders should enable them to seek responsibility in a self-directed way [McG60]. Other models describe the influence of generalized factors on motivation: *Motivation-Hygiene Theory* distinguishes between *motivators* that increase a person's work satisfaction and involvement, while *hygiene factors* go unnoticed as long as they are satisfied, but cause dissatisfaction if they are not met [Her05]. In terms of models that are more specific to the workplace, *job characteristics theory* determines the motivating potential of a specific job to an individual by describing it using the five characteristics *skill variety*, *task identity*, *task significance*, *autonomy*, and *feedback*. In order to be motivating, a job has to score high in at least one of the first three factors, as well as provide for high autonomy and level of feedback [RJ15].
Motivation has been regarded as an important factor for satisfaction in the software engineering industry, although it is not easy to define and measure due to the complex nature of the activities involved [Bee+08; Fra+11; LB02]. Several studies have found motivators ranging from external factors such as rewards, task-related factors such as variety and level of challenge of the work, and team-related factors such as a sense of belonging, recognition, trust, and good relationship with colleagues [Bee+08; Fra+11]. With regard to team composition and management, these motivational theories should be taken into account when deciding which individual to assign to a specific team, role, or task, since the same position or work environment can be more motivating to one person than another.

2.2.2 Team Roles

Using personality measures such as the MBTI or FFT (cf. Section 2.2.1) for composing teams is challenging because their primary aim is to quantify an individual's personality rather than the interrelations between multiple personalities. For instance, an individual can choose to act differently in a specific group because the context requires a certain need to be filled [RJ15]. Therefore, some team models use roles or competencies to describe the configuration of a team.
Margerison and McCann [MM90] distinguish between nine core competencies that a team should have: *Advising* (gathering and reporting information), *Innovating*

(experimenting with new ideas), *Promoting* (presenting opportunities), *Developing* (testing new approaches), *Organizing*, *Producing*, *Inspecting* (controlling working systems), *Maintaining* (upholding processes and standards), and *Linking* (coordinating with others) [MM90; Mar01]. Each of these competencies has a corresponding role that needs to be filled in a team. Their recommendation is that "the team members between them [should be able to] cover all the areas and have all-round team competence" [Mar01], although each member does not need to be equally skilled in all areas.

Table 2.1: Belbin's team roles, based on [BA09; Bel12]

Role	**Description**	**Strengths**	**Weaknesses**
Plant	Solves difficult problems creatively	Creative, imaginative	Ignores details, ineffective communication
Resource investigator	Develops contacts and explores opportunities	Enthusiastic, communicative	Overoptimistic, quickly loses interest
Co-ordinator	Promotes decision-making and focuses on objectives	Good at delegation, mature, confident	Prone to delegating too much, perceived as manipulative
Shaper	Focuses the team and ensures it keeps its momentum	Dynamic, challenging, courageous	Prone to provoke others or hurt people's feelings
Monitor Evaluator	Lends impartial judgements without emotions	Objective, strategic, sees all options	Lacks inspiration, overly critical
Teamworker	Facilitates cooperation and communication	Cooperative, perceptive, diplomatic	Influenceable, indecisive
Implementer	Turns ideas into actions through a concrete strategy	Reliable, efficient, conservative	Can be inflexible
Completer Finisher	Provides the final polish to the team's work	Conscientious, attention to detail	Inclined to too much work, nit-picky
Specialist	Has in-depth knowledge in an important area	Dedicated, single-minded	Narrow horizon, overlooks big picture

The Belbin Team Role Inventory[1] also lists nine team roles; each person can assume multiple roles, but all of them should be covered for the team to perform well [BA09; Bel12]. The team roles *Plant*, *Resource Investigator*, *Co-ordinator*, *Shaper*, *Monitor Evaluator*, *Teamworker*, *Implementer*, *Completer Finisher*, and *Specialist* are clusters of behavior that an individual can take on in a team environment. They are shown in Table 2.1 along with their respective strengths and weaknesses. The inventory provides specific recommendations for each role, both concerning how to work with others and how to improve one's own skills. Belbin's Team Roles have been extensively studied [ASS07] and have also been used as basis for a team composition model [WWB07].

2.2.3 Group Development Theory

Group development refers to the "changes through time in the internal structures, processes, and culture of the group" [SG74]. A variety of group development theories exist, and some researchers have categorized them according to their characteristics. Ven and Poole [VP95] categorize theories of organizational change according to the two dimensions *mode of change*, which can be prescribed or constructive, and whether the *unit of change*, which can be a single entity or multiple entities. According to Smith [Smi01], change within a group takes place along three dimensions: *Social* change refers to the evolution of the group's structure and the roles within, change in *activities* encompasses the tasks and processes, and the dimension of *culture* describes change in the group's norms, values and purpose. McGrath and Tschan [MT04] distinguish models which describe the group's stages of development as a unit in general, while others characterize its performance and are therefore dependent on the specific task that the group is carrying out.

In the following, we use the categorization by Smith [Smi01] who divides models of group development into three categories: *linear progressive models*, *cyclical & pendular models* and *non-phasic or hybrid models*. We shortly describe these categories and present one concrete model for each.

Linear progressive models assume that groups develop according to a pre-determined series of phases or stages, transitioning from one to the next in a linear way. These models are comparably easy to understand and have therefore been widely used by researchers. The models in this category are similar regarding the phases they describe: They all contain a stage in which the group is formed, followed by a period of conflict, after which norms and trust develop [Yuk13]; the group can then work productively until it dissolves [Smi01].

[1] `https://www.belbin.com/about/belbin-team-roles/`

The most widespread in this category are *Tuckman's stages of small group development*, a five-stage model through which groups transition regardless of the task at hand. Table 2.2 shows the processes that typically take place in each stage as well as the characteristics of group work and the emotions likely to be present in the group.

Table 2.2: Tuckman's stages of small group development, based on [Bon10; HA08; Map88; RJ15; TJ77].

	Processes	**Characteristics**	**Emotions**
Forming	Orientation towards others, orientation to the task, exchange of information	Uncertainty, courtesy, commitment	Confusion, caution
Storming	Competition, disagreement over task and processes	Conflict, confrontation, criticism	Dissatisfaction with others, conflict
Norming	Development of group structure, establishment of roles	Cohesiveness, clarification, support	Harmony, involvement
Performing	Working on the actual task, emphasis on productivity, emergence of solutions	Performance, creativity, decision-making, task orientation	Decreased emotionality, achievement
Adjourning	Task completion, reduction of dependency, wrapping up the project	Disintegration, consensus, compromise	Recognition, satisfaction, regret, closure

Tuckman's model is widely used for academic studies of team building and team performance (cf. e.g. [KM16]), but several limitations have been identified, such as the fact that it was developed based on observations of therapy groups, which limits its generalizability. Moreover, the group is regarded as a closed system without outside influences, and the model does not provide for the group to transition through the stages in a non-linear fashion, e.g. transition back and forth or never move beyond a particular stage [Bon10].

Cyclical & pendular models address the limitations of linear models in that they allow groups to revisit stages or iterate back and forth between stages as they address challenges that can be due to external factors, a change in the group's composition or a change in their task. While the stages described are often similar to linear models, this category is more advanced and realistic in terms of viewing the group as "able to

assess their external and internal environments and make either subtle or dramatic changes that allow them to accommodate the changes they face", making them more dynamic entities [Smi01]. A commonality of the models in this category is that it is not relevant in which order or how often a group visits the stages, but in order for it to reach its full potential, it has to deal with the challenges of each stage.
An example cyclical model is the *AGIL scheme* described by Hare [Har73], which describes the stages of *Latency (L)*, *Adaptation (A)*, *Integration (I)* and *Goal-attainment (G)*. The definition of the group's purpose takes place in the *Latency* stage, after which new skills required for the task are acquired (*Adaptation*), the group organizes itself so that they become self-sufficient and less dependent on the leader (*Integration*) and finally the group can work on the task (*Goal-attainment*). When the group dissolves, they transition back to the *Latency* stage and redefine their relationships for the future. The model does not define a fixed duration or progression of the phases and claims that it is dependent on the leader of the group as well as the skills and personality of its members. It is also possible for sub-groups to go through the stages individually or carry the rest of the group with them to the next stage [Har73; Smi01].
Non-phasic or hybrid models go beyond the other two categories in their degree of freedom as to how the group develops. These models do not prescribe a pattern, but are rather contingency models which assume that the development of the group mostly results from environmental factors, seeing groups as "open systems which are embedded in larger organizations and environments" [Smi01]. We shortly present two examples to illustrate the nature of this category.
The *TEAM model*, shown in Figure 2.6, combines Tuckman's stages with ideas from some of the cyclical models. The stages from Tuckman's model are slightly extended and describe the approximate progression of the group, except that it no longer has to be linear and a group can also start at a more advanced stage depending on their prior experience. In addition, the model describes two individual activity tracks that run through these stages. The activities shown in green in the upper row are specific to the task performed by the group, such as interactions with domain-specific tools and technical aspects of the task. With these activities, the group can be expected to become more proficient as they progress. The bottom row shows the orange activities which are generic team skills referring to the interaction and relationships of the team members. As the group gains maturity, these two activity tracks converge to signify productivity and group cohesion.

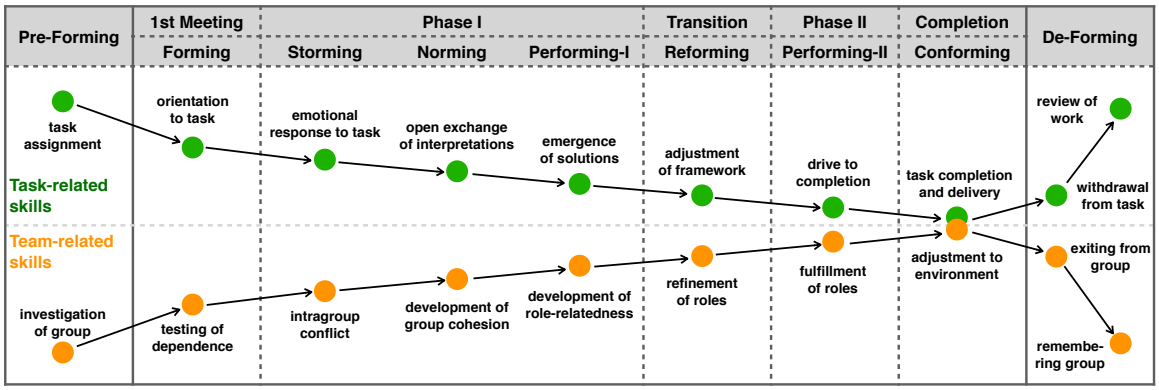


Figure 2.6: Overview of the TEAM model, simplified from [MSG93]

2.2.4 Algorithmic Team Composition

Many researchers agree that team composition is a complex problem that quickly becomes hard to solve using a manual approach, especially with a large pool of potential team members or a growing number of variables considered [Bos+11; FPC01; LLOR10; SG10; YC11]. Therefore, related work exists on mathematical models, algorithmic approaches and tools to overcome this challenge. Part of the team composition research is focused on assigning students to teams for applied courses. This section summarizes existing approaches to algorithmic or algorithmically supported team composition from both the educational field and industry.

Henry [Hen83] describes the algorithmic composition of teams in a beginner software engineering course without the involvement of industry partners. This approach uses information about previous computer science courses and grades, preferred teammates and information about time commitment, such as the amount of current courses and whether the student is busy with job interviews before graduation. From this data, they calculate a score "reflecting the amount and quality of work each student is capable of producing" [Hen83] in order to create teams that are above a specified threshold.

Team-Maker is a tool that allows instructors to define an individual set of questions to be answered by students, based on which the teams are assigned [LLOR10]. Instructors can weight each question according to its importance and decide whether the teams should be balanced or created based on similar answers to that question. The tool uses the hill climbing algorithm, i.e., it starts with a random assignment and calculates how well each variable has been met and how well the overall assignment satisfies the given criteria, iteratively adapting the result to maximize its quality. It also takes into account minorities by deprioritizing teams that have a single underrepresented member. The instructor is presented with an optimal solution accompanied by summary statistics and graphics so that they can adapt the input variables and run the algorithm again

if they are not satisfied with the result.

In terms of mathematical models for team composition, Strnad and Guid [SG10] propose a single-objective method maximizing utilization of skills in the resulting teams. Their approach uses flat staffing of multiple teams at once and allows the specification of skills in a flexible way: for instance, one can choose whether a set of skills can be covered by a single person or has to be provided by multiple people. Skills and project requirements are represented using fuzzy logic as opposed to binary variables. Yang and Chou [YC11] use a multiobjective approach, arguing that the problem is too complex for a single objective. Their model uses particle swarm optimization, an approach that claims to be efficient even with multiple objectives and a large solution space. They maximize profit and utilization while minimizing overtime and variation of tasks. However, they assign already existing teams to incoming projects as opposed to putting individuals into project teams.

While the above-described approaches do not explicitly address interpersonal relationships and factors that are important for teamwork, other researchers concentrate solely on these variables. For instance, Chen et al. [CCFC12] propose a nonlinear optimization problem using the *Genetic Algorithm* to assign teams based on the candidates' preferences concerning potential teammates as well as the popularity of the candidates among their peers. Other research concentrates on composing teams with favorable personality profiles, e.g. the creation of balanced teams according to Belbin's Team Role Inventory (cf. Section 2.2.2) [LPM09; PSCB00; WWB07]. Chen and Lin [CL04] combine multi-functional skills of the candidates with personality profiles to assign teams that are likely to have good working relationships. However, their algorithm uses pairwise comparisons of potential candidates instead of absolute ratings of their skills and they admit that there is no solid theoretical foundation of their assumptions as to how well the different personality profiles can work together [CL04].

A challenge when algorithmically composing teams is the presence of uncertain or incomplete information. The method by Strnad and Guid [SG10] described above uses fuzzy logic to represent project requirements. Tseng et al. [THCG04] also use a fuzzy approach for modeling requirements and skills and combine it with grey decision-making to find an optimal solution under ambiguity.

In terms of the application of algorithmic team composition in industry, we found only a few cases reported in the literature. For instance, the IBM Institute for Business Value presents the application of IBM Watson technology in a Japanese engineering consultancy to match their employees to positions in their clients' projects. The approach uses cognitive computing, including natural language processing on project and employee data as well as unstructured search in company-internal files to determine the most suitable candidates, taking skills, personality, and cultural fit into

account [IBM16]. In an interview study with practitioners from technology consulting companies, we found that in the majority cases their process involves using a tool to search for employees with specific skills, but they find it very challenging to document skills with sufficient granularity, especially with new or fast-evolving technologies. Interestingly, the majority of interviewed practitioners expressed the opinion that an algorithmic solution would yield better results than the current process, but they were worried about privacy and trust aspects [DB18b]. The results of this interview study are described and discussed in further detail in the next section.

2.3 Team Composition in Practice

This section presents the current state of team composition in practice elicited from an interview study with practitioners. We conducted semi-structured interviews with employees of technology consulting organizations who have been involved in team composition at least on a weekly basis for a minimum of two years. In each interview, we elicited a description of the team composition approach in terms of process steps, roles involved, and available tool support. Details on the approach and the participants can be found in the corresponding publication [DB18b].
Section 2.3.1 presents the formalized project stages with regard to requirements for team composition elicited from the interviews, followed by the typical roles involved in the process in Section 2.3.2. Finally, Section 2.3.3 describes and discusses three types of processes for team composition that were observed.

2.3.1 Team Composition and Project States

While the interviewed practitioners described processes of varying level of formalization and roles involved, their responses can be abstracted to reveal common states of a project that differ regarding the role of team composition. The identified states are shown in Figure 2.7 and further explained in the publication [DB18b].
`Project Preparation` refers to a state in which the project is being planned but work has not started yet. If the organization performs work for external clients, they often have to place a bid in a tender, for which they receive the project requirements and the most important constraints such as the time frame of the project from the client. At this stage, budget considerations are important to be able to place a bid. Thus, the team's initial composition is determined in the form of a team template involving generic roles and the required seniority of employees filling those roles. Some interviewees said that they assign the project leader or senior architect at this stage already.

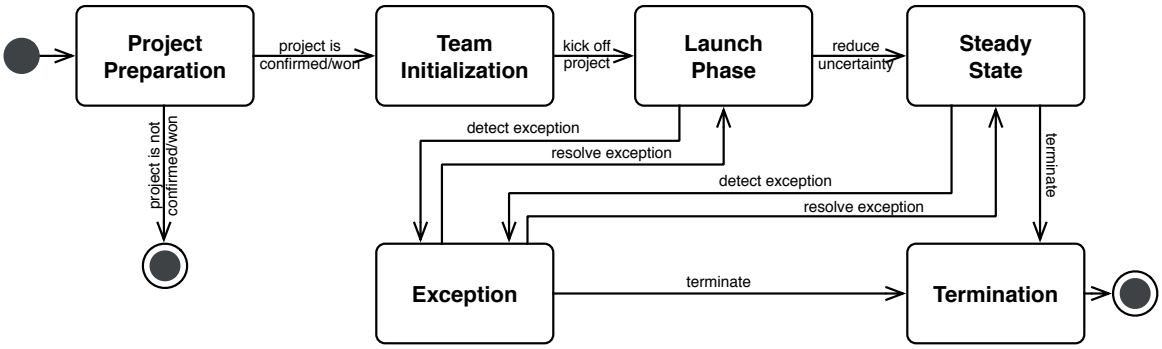


Figure 2.7: Project states with regard to team composition

If the project is won or confirmed by the client, it transitions to the `Team Initialization` state. At this step, concrete people are assigned to the roles of the team template using information available about the project and the possible team members, thus creating the initial team composition. Interviewees described tools for searching for employees with a specific skill or experience in a certain industry, but they unanimously said that obtaining the required information from a tool was challenging, which is why informal channels are the preferred route if available.
The project transitions into the `Launch Phase` when it is kicked off. In this state, the team works on reducing uncertainty in terms of requirements, processes and tools as well as communication. Once the team has successfully reduced the uncertainty, the project is in a `Steady State` and often remains there until its `Termination`. During these states, team composition is monitored.
In the scope of this research, an `Exception` state occurs if an event requires an unplanned change in the team's composition. During the `Launch Phase` or `Steady State`, this can be due to a team member leaving or a change in requirements that leads to a lack of skills on the team, for instance. A transition to a non-exception state is possible once the issue is resolved, but the project can also be terminated directly from this state if the problem cannot be fixed.

2.3.2 Roles Involved in Team Composition

The analysis of the interview study results revealed three process types. The following roles were observed in all process types:

- `Employee`: Member of the organization who is part of a pool of possible team members; sometimes includes external persons such as freelancers.
- `Project Leader`: Experienced Employee and managerial lead of the project; sometimes involved in team composition.

- `Staffer`: Decides whether an Employee is assigned to a project; has knowledge about a number of Employees' skills, availability and interests.

Figure 2.8 shows how the actors are symbolized in the following figures depicting the three types of processes observed.

Figure 2.8: Actors observed in team composition processes in industry

2.3.3 Team Composition Process Types

This section presents a formalization and categorization of team composition approaches observed in project-based organizations in the technology industry during the course of the interview study with practitioners. We describe the three identified types *Decentralized Process*, *Centralized Process*, and *Networked process* and then compare them to each other with regard to the number of people with personal knowledge of employees as well as the importance of individual relationships.

Decentralized Process

We call the most-described type of team composition approach the *Decentralized Process*. Half of the interviewees, all of them members of matrix organizations, described a variant of this approach, which is visualized in Figure 2.9. In this process, team composition is supervised by the future `Project Leader`, sometimes accompanied by a senior architect or advised by people who were involved in analyzing the project requirements before the project was won by the organization. The `Project Leader` talks to one or multiple `Line Managers` (e.g. department leads). Each of them assumes the role of Staffer and is responsible for a group of people whose interests, skills and experience they know well through regular personal contact. Interviewees estimated the size of this group to be between 15 and 40 people.
Practitioners also described defined mechanisms such as weekly meetings in which the Staffers update each other about their `Employees`' availability as well as current project shortages. In addition, three out of four organizations in this category have a database to document skills and availability which is updated by the `Employees` themselves. Although these tools are available, practitioners described challenges around documenting skills in a sufficient granularity and accuracy, which is described

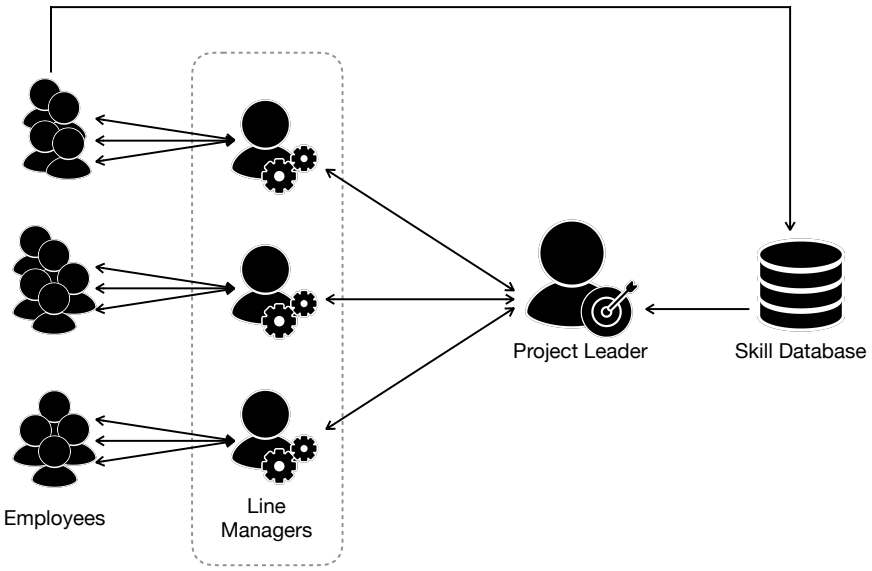


Figure 2.9: Decentralized Process for team composition

in detail in [DB18b]. Thus, tools are relied upon if personal knowledge is not available, for example if they are staffing a position from an offshore development center. We observed that if possible, practitioners prefer to work with people they have personal experience with, which gives them a better sense of whether this person fits the team and project at hand. In the *Decentralized Process* they do not contact the `Employees` directly, but talk to their `Line Managers` about their availability and potential for the position.

Centralized Process

The *Centralized Process* was described by interviewees of two large organizations (more than 20.000 employees). This approach features an `Assignment Advisor` role which is filled by one or multiple members of the HR department. The `Assignment Advisor` assumes the role of Staffer; their primary responsibility is to assist the `Project Leader` with team composition in all project stages, as opposed to the *Decentralized Process* in which the line manager's responsibilities are added to their regular project tasks. The `Assignment Advisor` has personal knowledge of a large amount of people (interviewees described a groups of 100-200 `Employees`). They use tools to keep current information about their `Employees`' availability, interests, and skills, but this information is not directly available to the `Project Leaders`. Practitioners described regular, e.g. weekly, alignment meetings with the `Project Leaders` to discuss current team member availability, possible shortages, and needs of future projects.

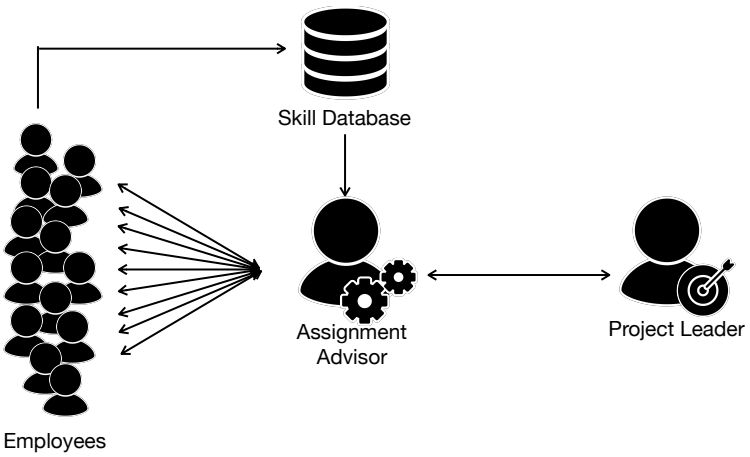


Figure 2.10: Centralized Process for team composition

Networked Process

Finally, a *Networked Process* requires `Employees` to interact directly and regularly with *Project Leaders* and Staffers. For example, one interviewee spoke of a sales team that acquires projects and releases their description and requirements to the whole organization. *Employees* are encouraged to build a wide personal network to the `Staffing Team` and to regularly inform them about their interests and satisfaction with their current project assignment. The `Staffing Team` meets regularly to collaboratively fill vacancies in current projects and initialize new teams according to their personal experience and knowledge of `Employees`. The two interviewees that described versions of this approach were members of smaller organizations with 500-1.000 employees.

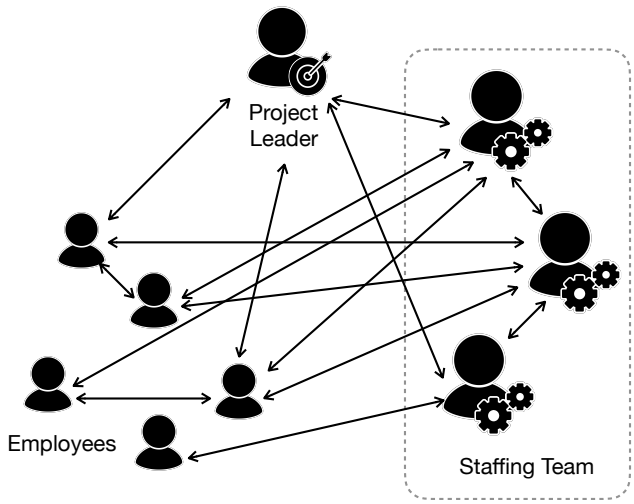


Figure 2.11: Networked Process for team composition

Comparison & Summary

Comparing these types of approaches to each other, the number of people with personal knowledge of employees is lowest in the *Centralized Process*, while the *Decentralized* and *Networked Process* have a potentially more fragmented picture of the pool of employees in the organization. Individual relationships are most important in the *Networked Process*, in which the organization relies on employee's willingness and capability of communicating their interests instead of relying on defined feedback mechanisms and supervision hierarchies.
As this categorization is based on interviews with a sample of only eight practitioners, it cannot provide an exhaustive picture of current team composition in practice. However, it gives a first insight into the types of approaches that exist and the communication channels and roles they involve, which is a source of requirements for tailoring the TEMPO framework.

Chapter 3

TEMPO Framework

> Software innovation, like almost every other kind of innovation, requires the ability to collaborate and share ideas with other people, and to sit down and talk with customers and get their feedback and understand their needs.
>
> Bill Gates

This chapter presents and describes TEMPO, a framework for team composition and management in project-based organizations.
We follow the Tornado model [BKW12], starting with the visionary scenarios of the framework in Section 3.1. Following this, Section 3.2 describes the functional and non-functional requirements for TEMPO, followed by use cases in Section 3.3. Section 3.4 presents the framework's problem domain objects in the Analysis Object Model, and Section 3.5 contains its dynamic model. Section 3.6 introduces patterns and objects of the solution domain in the form of the Object Design Model. Section 3.7 evaluates two types of algorithmic approaches to illustrate TEMPO's suggestion mechanism. Finally, Section 3.8 discusses prerequisites for using the framework as well as its limitations.

3.1 Visionary Scenarios

The visionary scenarios for TEMPO were derived from the observations from practitioner interviews explained in Section 2.3 as well as our experience from composing teams for multi-project courses described in [DAHB18]. The scenarios contain the main actors *Staffer*, *Project Leader* and *Employee*, which are capitalized in the description of the scenario.
To facilitate understanding, all but Visionary Scenario VI depict a continuous sequence of events in connection with the system, i.e., each scenario builds on the events described in the previous scenarios.

Visionary Scenario I: Prepare Project

Barbara, who is a Staffer at 'UnReality Solutions', a large matrix organization developing Virtual Reality applications for their clients, prepares to place a bid for a new project.
She reads through the material she received for the 'SalesKingdom VR' project, a state-of-the-art sales and customer relationship management application in virtual reality. The client sent over the visionary scenarios, high-level requirements and deadlines for major milestones for her to prepare a bid. In order to be able to make an offer, the resources and time needed to realize the project have to be calculated. The most important determinant for the project budget are the costs in human resources, which is why she needs to know now what kind of people are needed.
Barbara thinks that staffing this team will be a challenging task: she will need five developers with advanced skills in virtual reality technologies as well as an experienced senior software architect and two business analysts who are knowledgeable in sales processes. She specifies these requirements as constraints for the SalesKingdom VR project team in TEMPO. Barbara decides that the project can also start off with only

three developers with advanced VR skills if the other two are interested in learning more about this field – she presumes that if they are motivated, the others will help them get into the technologies quickly. She sets a lower bound of 3 and an upper bound of 5 for the developer positions and additionally configures an objective to maximize VR skills on the team. She thinks the team can make it with only one business analyst, so he specifies a lower bound of 1 and an upper bound of 2 for this constraint. The software architect has to be of high seniority due to the technical challenges in the software architecture, so she sets this as an exact constraint. Barbara has now configured a team template for the SalesKingdom VR project that contains some template positions, but has not assigned any concrete people yet.
As it is important for the project to start soon, Barbara consults with some senior Employees to see who can lead it. Choosing a Project Leader now will enable them to prepare and hit the ground running. She talks to Alex, an experienced Project Leader who has worked in the sales industry before, and they decide that he will assume this role if UnReality Solutions wins the tender. She staffs the team template position of Project Leader with Alex so that other Staffers know that he might not be available once the project starts. Barbara uses the costs determined by the team template to place a bid in the tender for SalesKingdom VR.

Visionary Scenario II: Initialize Project Team

After the organization has won the tender for SalesKingdom VR, Alex, the future Project Leader, and Barbara, a Staffer of the organization, assign the first people to the project. Alex reads through the requirements received from the client. He discusses them with Barbara and they conclude that the previously set constraints and objectives still hold.
By now, it has been decided that the project will be developed for the HoloLens VR glasses, so they add an additional objective in TEMPO to have as many developers as possible with skills not only in VR, but also in this specific technology. Alex knows a software architect from a previous project that he would like to work with again, so he makes a call to see if he would be interested. Barbara checks his availability and then assigns him to the team as a fixed member. They set the project start to be one month in the future, and the duration to nine months as specified by the contract with the client. They are now satisfied with the constraints and decide to generate the new team.
TEMPO generates three suggested teams from the pool employees of the organization who satisfy the specified criteria. Alex and Barbara check the suggested teams. The first suggestion, which TEMPO indicates matches their criteria best, seems to be the most suitable: The second suggestion has only four developers and the third suggestion

can start in a month earliest, and they are hoping for an earlier project kickoff. Alex and Barbara select the first suggestion for further refinement.
Alex likes the five suggested developers, but Barbara can see that both business analysts are in a senior position. This means higher costs for the budget and from her experience can lead to an unclear division of responsibilities. They use TEMPO to look for a junior business analyst with either existing skills or interest in sales applications. Using the pool of Employees of the organization, TEMPO suggests business analysts with the specified criteria. They find Albert, who Alex has heard good feedback about from a colleague. Thus, they exchange one of the senior business analysts with Albert. TEMPO warns Alex and Barbara that this team member is not available for the whole duration of the project, only during the first three months. Since this period is the most intense part for analysis activities, Alex thinks he can work with only one analyst afterwards and dismisses the warning.
TEMPO further indicates that the prospective team's personality configuration is favorable. Since almost none of the future team members have worked together before, team coaching and team building activities are recommended. Alex makes a note to set up a coaching session. Both Alex and Barbara are satisfied with the project team composition. Barbara exports the results to check them with the human resources department, the new project team members and their managers.

Visionary Scenario III: Monitor Team Composition

The SalesKingdom VR project has been running for three months and Alex, the Project Leader, opens TEMPO for his regular check-up on the team's composition. He knows that Albert, the junior business analyst of the project, is leaving at the end of the month, and he wants to see how this affects the team composition. He opens TEMPO and sees the projects he is responsible for. The SalesKingdom VR team already shows a warning.
TEMPO alerts Alex that there is a significant deviation from the previously defined team composition criteria: The second business analyst position that was specified during project preparation will no longer be staffed after Albert's departure. Alex is confident that the project can continue with a single business analyst. He dismisses the warning and tells TEMPO not to show it again.
TEMPO further signals that there are changes in two of the developers' interests that possibly affect the team composition: Lisa has indicated that she is no longer interested in working in projects in the sales industry and Robert has expressed a wish to be involved in another upcoming project. Alex thinks back and finds that over the past two weeks, Lisa seemed demotivated and has not contributed as much as before to the team. He quickly checks a metric that he has previously set up, which imports

some data from the code repository and shows the development activity, and sees that Lisa has also contributed less to the code base than her fellow team members. Alex talks to Lisa and after confirming her wish to leave the team, he looks for a suitable replacement using TEMPO's search functionality.
Since Robert has stated that he would like to be involved in an upcoming project, Alex talks to him and agrees that he will split his work time evenly between both projects. Alex adapts Robert's availability for SalesKingdom VR to 50% and decides that his team is adequately staffed for the moment.
Alex verifies that his adapted team's interest level for the project and its technologies is sufficiently high. He exports the resulting team composition to set the changes in motion with all involved parties.

Visionary Scenario IV: Provide Employee Information

Robert, who has been an Employee at UnReality Solutions for just over a year, opens TEMPO to update his information. He sees the information he entered when he started his job. Since then, he has gained a lot of experience in embedded development through several open source projects he has contributed to in his free time. He therefore sets his skill level for embedded development from 'beginner' to 'advanced' and describes the projects he has worked on.
While working on the open source projects, Robert's interest in embedded development has grown. He sets his level of interest from 'medium' to 'high'. Since Robert has been working on the SalesKingdom VR team for almost a year, TEMPO suggests that he adds the HoloLens to his list of skills. He accepts the suggestion and describes the work he has been doing as part of the project in more detail.
The work in the SalesKingdom VR team has been a pleasant experience, but Robert would like a new challenge. He has heard about a project with the Edison chip from a colleague and is extremely interested in being a developer in that future team. He browses through the organization's upcoming projects in TEMPO and finds 'IntelliGent' by its description. He indicates a 'very high' interest level for that project.
Robert verifies that his remaining skill and interest levels are correct and saves the data. He chooses to publish the information to every department in the organization, as he wants all colleagues or managers to be able to access it when they are looking for new team members.

Visionary Scenario V: Refine Team Composition Criteria

The SalesKingdom VR project has finished and the software has been accepted by the client. During the project retrospective, multiple team members mentioned that while they think their team had the required skills to realize the product and the result was satisfactory, there were some interpersonal issues in the team. While three of the developers stated that it was difficult for them to understand how the more introverted team members communicate, the rest of the team complained that the extroverted team members were often too loud and stressful in their way of solving conflicts.
Since the same concerns have occurred in multiple projects, Alex decides that factors of personality should be taken into account for composing future teams. He speaks to Barbara, who has heard this argument from several other Project Leaders. She configures TEMPO to take the team's personality profile into account when generating suggestions for future teams, as opposed to just visualizing it like before.
Another issue elicited from the retrospective was the fact that only one team member did not speak German, resulting in him being excluded by the rest of the team when they did not consistently communicate in English. After Alex tells her about this, Barbara decides to alert Project Leaders of this in the future: She uses TEMPO to configure a warning when the German skill level of a single team member is of value 'beginner' or 'none'.
Barbara saves the new criteria, which TEMPO will take into account for future teams and also apply to the monitoring of current teams.

Visionary Scenario VI: Initialize Multi-project Organization

Tim has to staff the 11 projects of the software hackathon that takes place next week: All 86 people who signed up will be assigned at once to exactly one team based on similar global constraints.
Tim wants to create teams of similar size, so he uses TEMPO to set up a global constraint for the team size with a lower bound of 7 and an upper bound of 8. He then remembers that CoolCompany is the main sponsor of the hackathon, which is why they were promised a larger team. Tim sets up a team-level constraint for 9-11 team members. In order to create fair circumstances across teams, Tim decides to distribute participants with a high level of software engineering skills equally across all projects. Moreover, he defines an objective to maximize interest in the project and its technologies and doubles its weight since he wants to ensure that assigned team members are extremely motivated to work on their projects.
TEMPO generates two possible assignments for Tim. All of the previously specified constraints have been satisfied by the algorithmic initialization for both of them. He

picks the one with the higher overall score in terms of objectives and reviews it. Tim remembers that the client of the CoolCompany team indicated that he would like to work with Sam. He is currently assigned to another team, but has indicated a high level of interest for CoolCompany. Hence, Tim moves Sam over to this project.
TEMPO warns Tim that the CoolCompany project now exceeds the specified limit of 11 members and also has more skilled members than the average. Tim moves one of the highly skilled team members to another project which is not overstaffed. He continues to make adjustments, receiving feedback on his actions from TEMPO. Once Tim is satisfied with the assignment, he exports it to inform the hackathon participants.

3.2 Requirements

As suggested by the Technology Transfer Model [GGLW06] and described in Section 1.2, the problem formulation and requirements for the framework are derived from the needs observed in practice. The requirements for TEMPO are elicited from the visionary scenarios described in Section 3.1.
Section 3.2.1 contains the functional requirements (**FR**s) for TEMPO, which "describe the interactions between the system and its environment independent of its implementation" [BD09]. Section 3.2.2 contains the non-functional requirements (**NFR**s) for the framework.

3.2.1 Functional Requirements

The functional requirements are structured based on the project states explained in Section 2.3.1. We also keep the roles of Staffer, Employee and Project Leader identified in Section 2.3.2.

Project Preparation

The majority of interviewed practitioners described a *Project Preparation* stage in which they analyze requirements and plan which resources are needed in order to determine whether the project is feasible for the organization, and if this is the case, how much they should charge for its realization (Section 2.3.1). This is done by the future project leader or a person with a role dedicated to defining projects. At this stage, the team is determined in terms of number and seniority of employees, and sometimes key positions such as the project leader are already staffed.
In addition to defining the project, information about employees needs to be available to be able to staff the team later on. The interviewed organizations use self-reported information which in some cases is extended using experiences of their manager or

data from previous projects the employee has worked on. This step is described as challenging by the majority of practitioners, especially regarding the documentation of skills at a sufficient level of granularity when it comes to fast-evolving technologies [DB18b].

FR1 The Staffer can configure a team template without assigning concrete people to the project yet. Some template roles can be pre-assigned at this stage.

FR2 The Staffer can define constraints and objectives for the project team. Constraints can have a lower or upper bound or both, while objectives are maximized or minimized. Both can have weights according to their importance.

FR3 Employees can provide their skills and interests. TEMPO makes suggestions for skills based on previous projects they have been a part of.

FR4 Employees can specify restrictions such as an inability to travel or a long absence in which they are not available for projects.

FR5 Employees can publish the information to specific groups in the organization.

Team Initialization

Once the organization wins the project, e.g. through a tender, the team template is staffed with concrete people, thus determining the initial team composition (Section 2.3). The interviewees described processes with varying degrees of formalization and tool support, and while some claimed that they have many choices of possible team members e.g. in offshore locations, others feel lucky if they find a single suitable fit [DB18b]. Therefore, the requirements in this section are based on commonalities observed in the interviews, such as the importance of collaboration and informal networks as well as the appreciation of motivation to learn over existing experience. We focus on extensibility and adaptability in the non-functional requirements in Section 3.2.2 to ensure that TEMPO can be tailored to the process of the organization.
Based on the learnings from the interviews and our experience from composing teams for project courses, algorithmically supporting the process would enable the project leader to take more criteria into account than it is possible with a purely manual process. However, the goal of TEMPO is not to run a fully algorithmic team composition – instead, this approach aims to find a way of algorithmically supporting the practitioner while taking their knowledge and experience into account and to thus achieve a combination of human intuition and algorithmic efficiency.

FR6 The Staffer receives suggestions of possible teams that fit the previously specified constraints and objectives.

FR7 The Staffer can select and manually adapt a suggestion. When adapting the team, the Staffer gets feedback regarding the effects of the change on the previously specified criteria and receives a warning if the quality of the composition, measured by the previously set constraints and objectives, deteriorates.

FR8 The Staffer can search for Employees based on the information available and manually assign them to a team.

FR9 The Staffer can initialize the team even if there is no previously defined team template.

FR10 The Staffer can initialize multiple teams at the same time.

Launch Phase & Steady State

During the *Launch Phase* and *Steady State* described in Section 2.3.1, team composition is monitored and adapted if needed. According to both the interviews and previously conducted empirical studies in our project courses, the most prevalent issues in a running project team revolve around decision-making processes and communication, including cultural issues and communication with offshore or non-co-located team members [DAHB18; DB18b; DB18a]. Other important factors that were named were team ownership, individual motivation and regular feedback through e.g. agile retrospectives. These factors can lead to a necessity to change the team's composition, but an adaptation can also be due to an employee taking on a different role or transitioning to another team: interviewees stressed the importance of being flexible and e.g. letting team members change projects in order to feel challenged and keep developing their skills [DB18b].
Based on the results, we argue that issues or changes arising during a running project are best handled through the intervention of an experienced manager. We envision TEMPO as a useful addition to the project leader's intuition by alerting them of irregularities in previously defined metrics to enable the early detection of issues.

FR11 The Project Leader can define metrics for a project as well as an acceptable range for each metric. If a metric is outside of the accepted range, TEMPO warns the Project Leader.

FR12 TEMPO allows automatic measurement of data through interfaces to development tools.

FR13 The Project Leader or Staffer can adapt the team composition by adding or removing team members at any time using the same process as during the team initialization.

Exception & Termination

TEMPO envisions an iterative refinement of the team composition criteria as the organization's process matures. Thus, it is possible to incorporate learnings from past projects to be taken into account for suggestions for future teams.

FR14 The Project Leader and Staffer can see the project's composition history in terms of changes in metrics over time as well as changes in the team composition.

FR15 Based on learnings from the project, the Staffer can specify global criteria for all future project teams.

3.2.2 Non-functional Requirements

This section describes TEMPO in terms of its non-functional requirements and classify them according to [Gra92].

NFR1 TEMPO should be tailorable to the organization's team composition process regarding the process steps, data, and roles involved (Adaptability).

NFR2 TEMPO should support the addition and modification of team suggestion algorithms (Extensibility).

NFR3 To adapt to organizations with strict privacy requirements, personal data can be securely deleted from TEMPO after generating suggestions (Security, Adaptability).

NFR4 Personal data available in TEMPO should be treated in a manner compliant to European privacy laws (Legal Requirements).

NFR5 TEMPO should be usable both on a device with a touch surface of different sizes (e.g. a tablet or a tabletop display) as well as on a device with peripheral devices such as keyboard and mouse (Usability).

3.3 Use Cases

Figure 3.1 shows the use case model for TEMPO. The use cases were derived from the requirements and scenarios described in Sections 3.1 and 3.2 as suggested by [BD09] and involve the actors Staffer, Project Leader and Employee as derived from the categorization of processes in practice in Section 2.3.2. Elements of the model referred to in the text are marked in `typewriter font`.

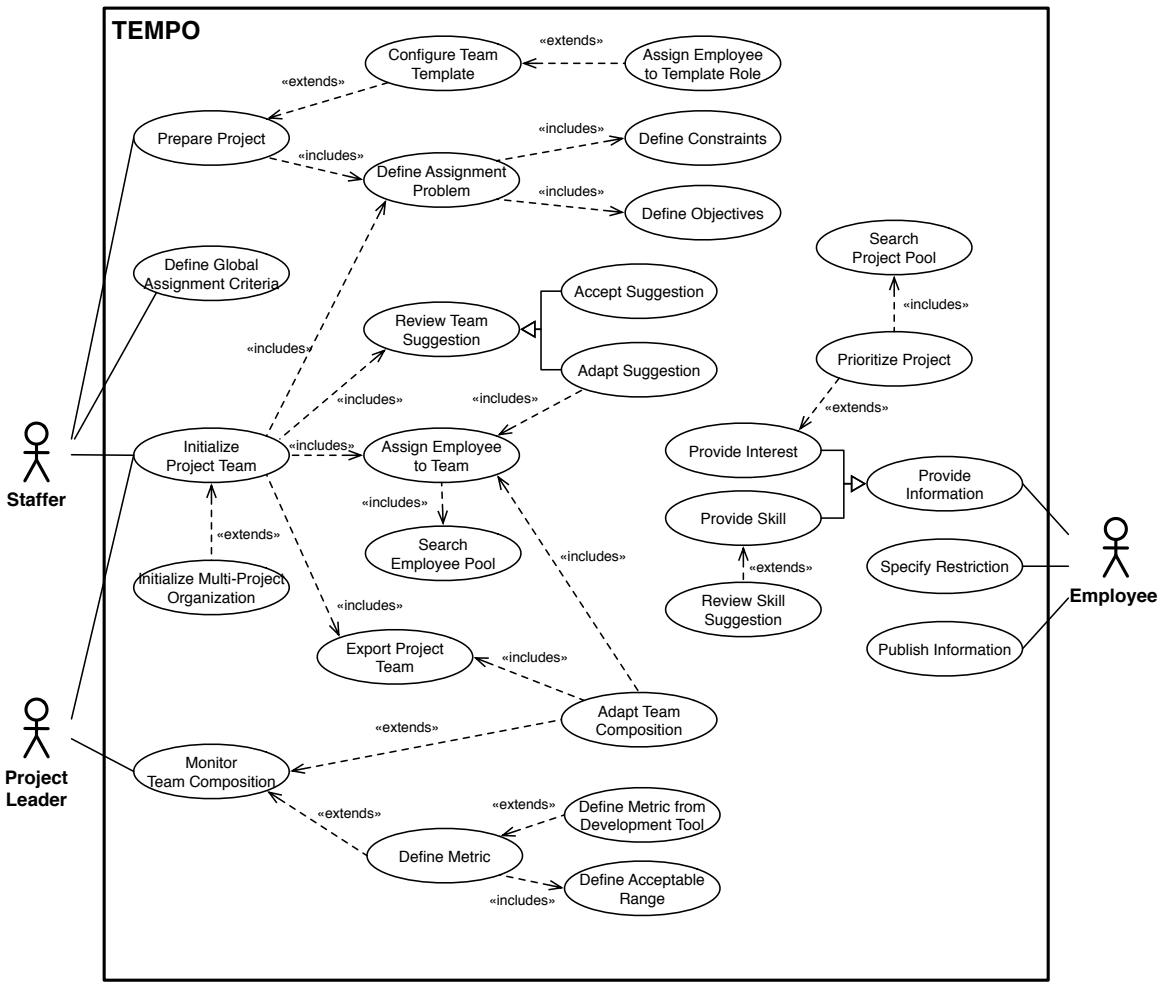


Figure 3.1: Use Case Model (UML Use Case Diagram)

The `Staffer` can either `prepare a project`, `initialize a project team` (possibly supported by the `Project Leader`) or `define global assignment criteria` based on learnings from previous projects. The `Prepare Project` use case was derived from Visionary Scenario I and can involve `configuring a team template` in the form of how many team members of which role are needed in the team to realize the project. If needed, the `Staffer` can `assign employees to template roles` at this stage already, for instance if a critical role should be staffed by a specific person. In order

to `define the assignment problem`, the `Staffer` should `define constraints` for rules that have to hold (e.g. the minimum number of people in a certain role on the team) and `define objectives` for goals that should be taken into account when providing suggestions (e.g. to maximize the number of people with experience in a certain sector).

When it comes to the `initializing a project team`, TEMPO provides suggestions for the team based on the previously defined assignment problem, which can also be re-defined in this step, as described in Visionary Scenario II. The `Staffer` and `Project Leader` can `review a team suggestion` and then decide to accept or adapt it. `Adapting a suggestion` involves changing the composition of the suggested team. In order to find suitable alternatives, TEMPO allows the actors to `search the employee pool` of the organization. When `assigning an employee to a team`, TEMPO communicates how this action impacts the previously defined constraints and objectives. Finally, when the actors are satisfied with the new team, they can `export the project team` for usage with other stakeholders or tools. The `Initialize Multi-Project Organization` use case is a special case of team initialization in which all employees are assigned to one project each at the same time, as described in Visionary Scenario VI.

During the course of a project, the `Project Leader` can use TEMPO to `monitor the team composition`. If the team members change, e.g. because a member leaves, they can check whether the remaining team satisfies the previously defined constraints and objectives. In case there is a need for action, the `Project Leader` can choose to `adapt the team composition` by e.g. adding a new member to the team. Similarly to the composition of a new team, he can `search the employee pool` and, once satisfied, `export the assignment`. In order to monitor the team composition, the `Project Leader` can also `define metrics` for the project so that they can be alerted by TEMPO if the metric goes beyond the `defined acceptable range` in the project. Visionary Scenario III describes a concrete instance of this use case.

TEMPO leverages the experience of the organization by offering to `define global assignment criteria`, making it possible to iteratively improve the framework's decision support. These criteria can be taken into account by TEMPO to pre-define future assignment problems or create warnings for teams in which the criteria do not hold true. Visionary Scenario V provides examples for both an adapted calculation and the configuration of warnings along with possible reasons for both.

In order to enable TEMPO to provide suggested assignments, the `Employee` needs to provide information about himself to the system. They can `provide information` in the form of `providing skills`, supported by TEMPO's suggestions based on previous projects they were part of in the organization. Information can also be given by `pro-`

viding interests, e.g. for technologies, fields, or the concrete prioritization of projects, which they can locate by searching the project pool. The Employee can also specify a restriction such as the inability to travel or a longer absence, which is taken into account by TEMPO when suggesting them for project teams. Once these details have been provided, the Employee can publish information to make it visible to specific parts of the organization. Visionary Scenario IV describes a concrete instance of this use case.

3.4 Analysis Object Model

Figure 3.2 shows the Analysis Object Model of TEMPO depicting the objects of the problem domain in the form of an UML class diagram. The objects are semantically divided into the three packages Assignment, Project and Employee, which are differently colored for better visibility.
An Organization has a ProjectPool consisting of multiple Projects, each of which has Requirements. Each Project is realized by a ProjectTeam which consists of multiple Employees that are its members. A Project can also have a defined TeamTemplate that consists of multiple Roles and defines the projected costs, but does not include concrete Employees yet.
Employees comprise the EmployeePool of the Organization. They are defined by their EmployeeInformation, which can consist of Skills, Interests, Personality and their ProjectHistory of previous Projects they have been part of. Moreover, they can take on one or multiple Roles in a project. An Employee can provide this information and publish it for others to see. They can also specify Restrictions such as absences with a start and end date.
An Employee can also be a ProjectLeader or Staffer, each of whom can perform different activities in TEMPO. For instance, the Staffer can prepare a project: As described in Section 3.3, this means that they define the AssignmentProblem in terms of its AssignmentCriteria that can be satisfiable Constraints with bounds or optimizable Objectives with given weights. The Staffer can also define global AssignmentCriteria which are used for all future AssignmentProblems.
Both the Staffer and the ProjectLeader can initialize a project team: TEMPO generates TeamSuggestions that are solutions to the given AssignmentProblem provided by the suggestion algorithm selected in its Solver. A TeamSuggestion is a complete configuration of the ProjectTeam in question, consisting of multiple Employees that can can be assigned to Roles defined in the TeamTemplate. It has a quality score depending on the weighted value of the Objectives of the solution, and can be reviewed or refined.

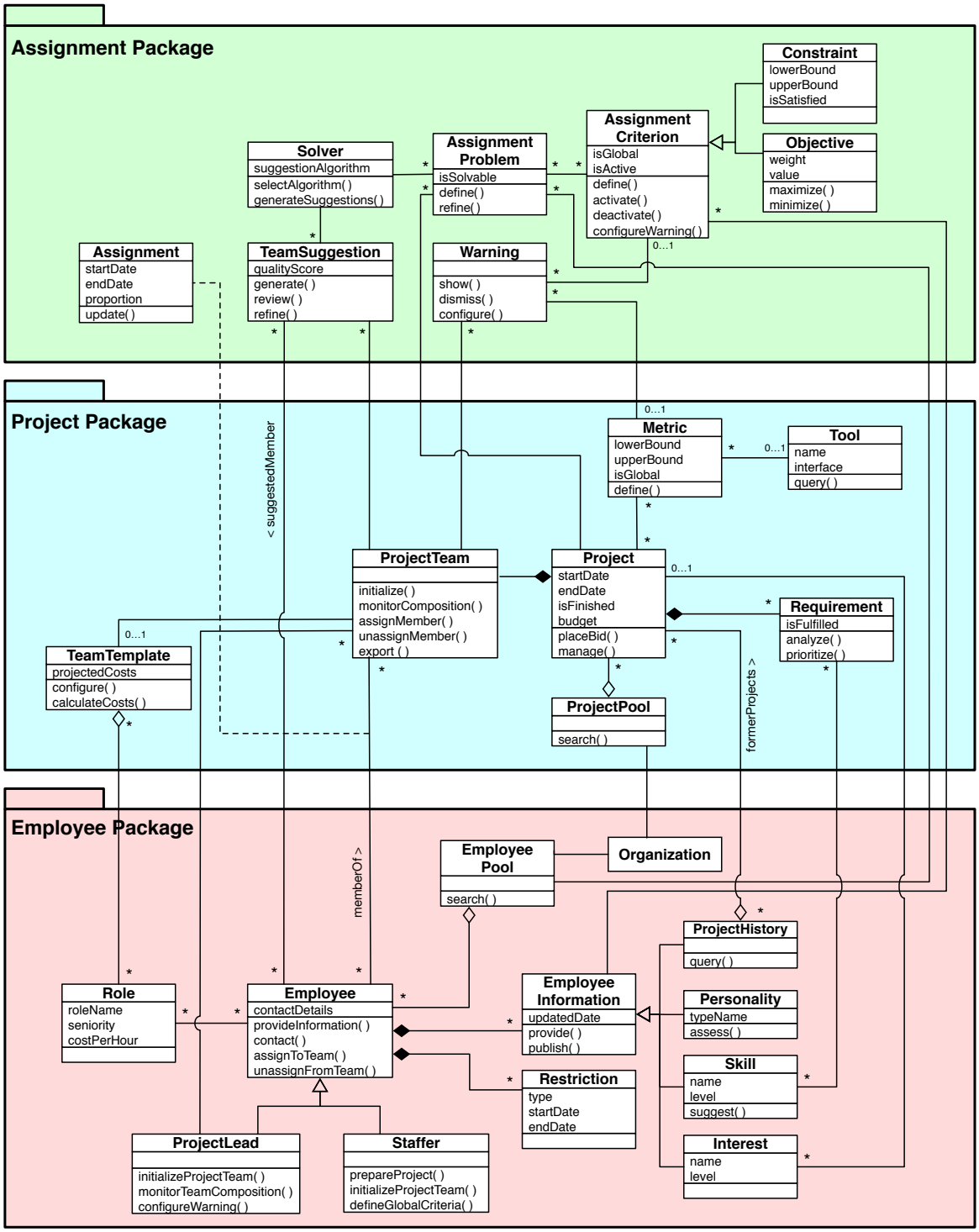


Figure 3.2: Analysis Object Model (UML Class Diagram)

The suggested `ProjectTeam`'s composition can now be adapted by assigning or unassigning members, possibly by performing a search in the `EmployeePool` using available `EmployeeInformation`. When an `Employee` is assigned to a `ProjectTeam`, the mapping is represented by the `Assignment` object. This object is represented by an UML

association class, which models an association with own behavior or state; although this type of class is no longer part of the UML 2.0 specification,[1] it is used here because it semantically suits this association in the model. An `Employee` can also be assigned to a `ProjectTeam` for a limited time or a limited proportion of their capacity. TEMPO provides feedback when refining the `TeamSuggestion`. If a change does not conform to the previously defined `AssignmentCriteria`, a `Warning` is shown.
The `ProjectLeader` manages the `Project`, which involves monitoring the composition of the corresponding `ProjectTeam`. They can define `Metrics` with bounds for the `Project`, which can be based on a query to a `Tool`. If a `Metric` is outside of the given bounds or the `AssignmentCriteria` no longer hold, TEMPO provides a `Warning` and enables the `ProjectLeader` to react.

3.5 Workflows

This section presents and describes TEMPO's dynamic model in terms of its workflows *Project Preparation*, *Team Initialization*, *Team Composition Monitoring* and *Employee Information*. These workflows have been derived from the scenarios and use cases described in Sections 3.1 and 3.3. An UML activity diagram is provided for each workflow and describe the progression of its activities.

3.5.1 Project Preparation Workflow

The *Project Preparation* workflow of TEMPO is shown in Figure 3.3. As described in Visionary Scenario I, this process can take place before the organization has won the project, although it can also be revisited before team initialization. When the organization has `received the specification` for the `Project`, the `Staffer` can `study and analyze its requirements` in the form of a first specification and decide if a team template is needed. If this is the case, they can `configure a team template` by `creating a role` and defining it in terms of its `minimum seniority`. If they have someone in mind, they can `assign an Employee to the template role`, resulting in an initial `Assignment`. The resulting `Team Template` is saved for future usage in TEMPO.
The `Staffer` also needs to `define the assignment problem`: If needed, they can `create constraints` and `specify their bounds` (e.g. number of members for a certain role should be within a range). They can also `create objectives` and for each of them `specify a goal` (i.e., should the objective function be maximized or minimized) and `specify a weight` referring to how important each `Objective` is

[1] `https://www.omg.org/spec/UML/2.0/About-UML/`

compared to the others. The created `Constraints` and `Objectives` define the resulting `Assignment Problem`.
If a `budget projection is needed`, e.g. to place a bid in a tender, the `Staffer` can `determine the budget` for the project using the `Team Template` they defined.

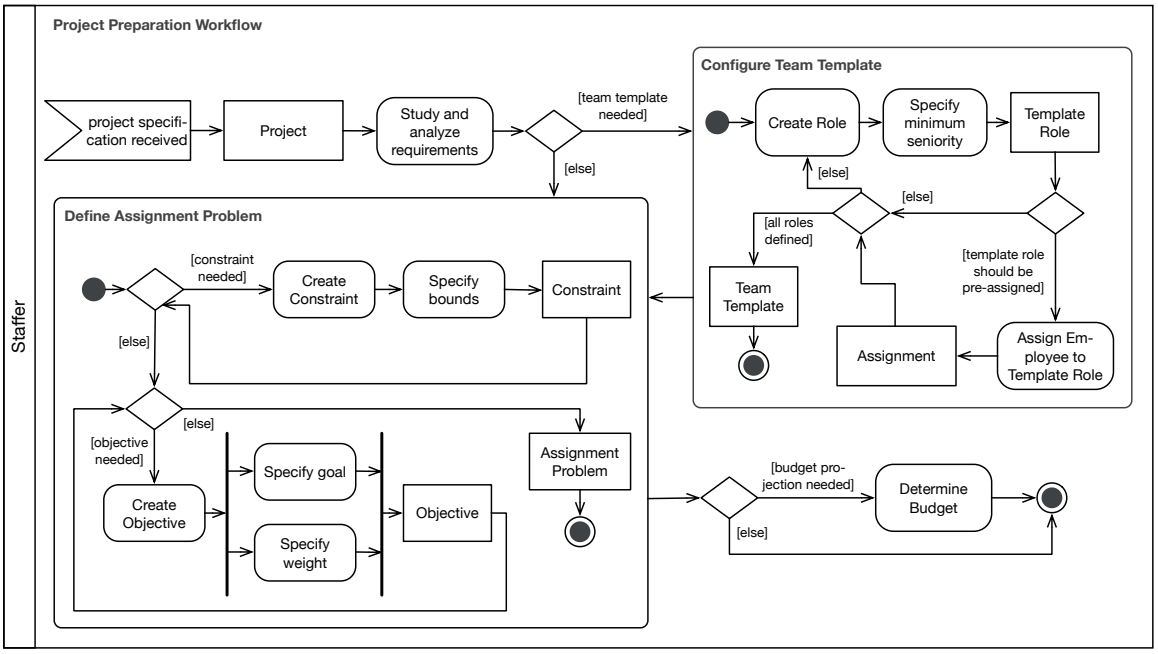


Figure 3.3: Project Preparation workflow (UML Activity Diagram)

3.5.2 Team Initialization Workflow

Figure 3.4 shows the *Team Initialization* workflow of TEMPO. An instance of this workflow has been illustrated in Visionary Scenario II. The workflow starts when the `project is ready to be staffed`, at which point the `Staffer` and/or `Project Leader` can `review the assignment criteria` defined before. If needed, they can `adapt the assignment criteria`, thus re-defining the `Assignment Problem`.
In case they have one ore more particular persons in mind who they want to have on this team, they can `pre-assign Employees to the Project Team`, resulting in initial `Assignments`. After this, they can use TEMPO to `generate suggestions` for this project team.
If the `Assignment Problem` is solvable, TEMPO provides one or multiple `Team Suggestions` which are algorithmically optimal solutions to the problem. The actors can `select a suggestion` and `review it`: If they find that it needs adaptation, they can adapt it by `searching the Employee Pool` for alternatives. If they find a suitable `Employee`, they can `assign them to the team`, resulting in another `Assignment`.

The `Staffer` and/or `Project Leader` can repeat this process until they are satisfied with the team, at which point they will `accept the suggestion`, thus initializing the `Project Team`. They can `export the Project Team` to e.g. forward the information to human resources and the newly created team members' managers.

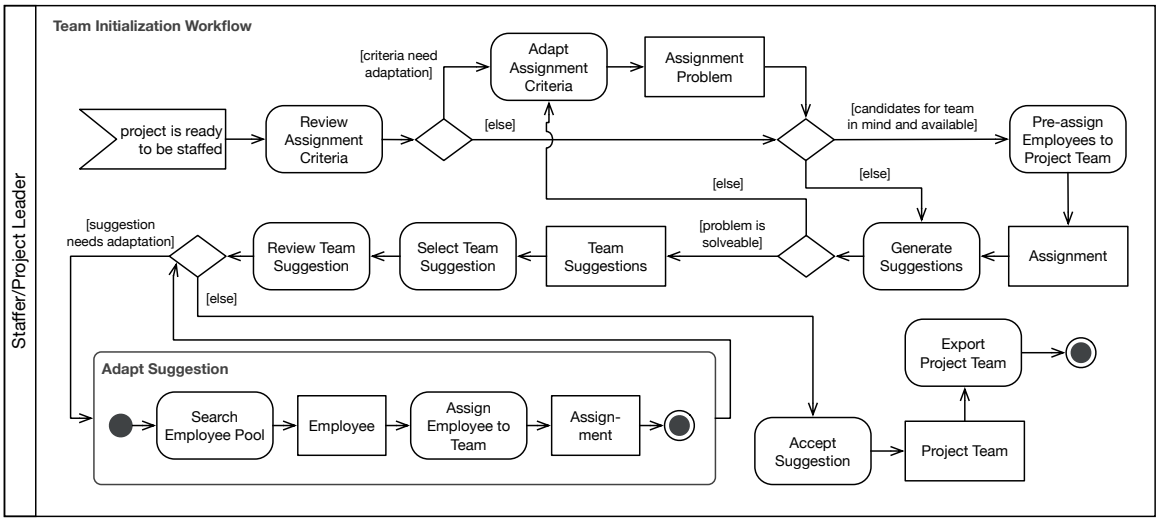


Figure 3.4: Team Initialization workflow (UML Activity Diagram)

3.5.3 Team Composition Monitoring Workflow

TEMPO's *Team Composition Monitoring* workflow, shown in Figure 3.5, has been previously illustrated in Visionary Scenario III.

The `Project Leader` begins this workflow either because he `received a project warning` from TEMPO, because the `team composition changes` or will likely change, or because they feel that there is a `metric needed` for the project. In the first case, he `reviews the warning` and can choose to `ignore the warning` if he decides that the team composition does not need to be adapted.

If the project team's composition needs to be changed, the `Project Leader` can `review the team and project state` in terms of how much it diverges from the constraints and objectives previously set. They can then `adapt the team composition` similarly to the team initialization workflow, resulting in an altered `Project Team`, and `export` it to make the necessary changes with other stakeholders. The `Project Leader` can also `adapt the assignment criteria` in terms of constraints and objectives to reflect the progression of the project, which results in an adapted `Assignment Problem`.

If a metric is needed, the `Project Leader` can `create a Metric` and along with it `define an interface to a development tool` if it should be measured using data queried from there. They also `define an acceptable range` for the `Metric`, and can choose to `create a warning` that is displayed by TEMPO when the `Metric` is

not in this range. This enables the `Project Leader` to be notified in case of issues and to react quickly by adapting the team's composition.

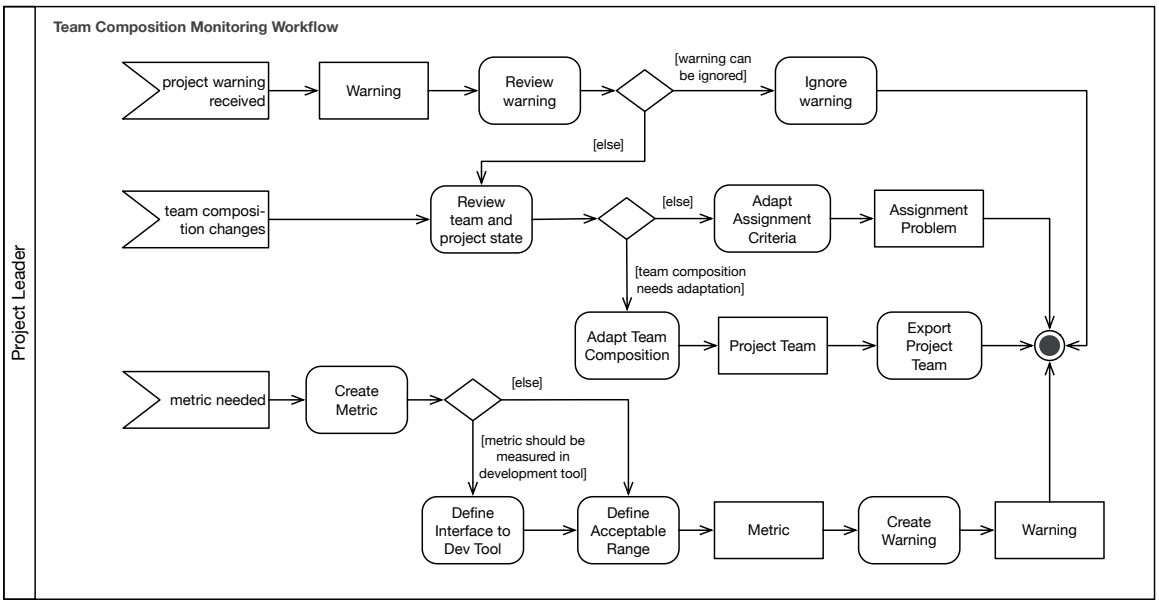


Figure 3.5: Team Composition Monitoring workflow (UML Activity Diagram)

3.5.4 Employee Information Workflow

As described in Visionary Scenario IV, the `Employee` needs to provide information that TEMPO can take into account for providing suggestions during project initialization. This *Employee Information* workflow is illustrated in Figure 3.6.

The `Employee` can `review existing information` and `adapt their information` if it needs updates. In case new information is needed, they can `provide a skill` or `provide an interest`. TEMPO supports the `Employee` by providing suggestions for `Skill`s based on previous projects they have been a part of. They can `review the suggestion` and either accept it and `specify their level of knowledge for the skill` or `create an entirely new skill`.

In the case of interests, the `Employee` can `create an Interest` with a corresponding `interest level`. For instance, it might be that they do not have experience with a particular technology, but are extremely interested in learning more about it. Alternatively, they can `search the project pool` and if they find a `Project` they are interested in, `specify their interest level`.

The `Employee` can provide as many `Skill`s and `Interest`s as needed, and if this `information should be visible` to other parts of the organization, they can `publish the information`. They can also `create a Restriction` such as a longer absence

or an inability to travel, and `specify the duration`, i.e., the time span during which it is valid and will be taken into account by TEMPO.

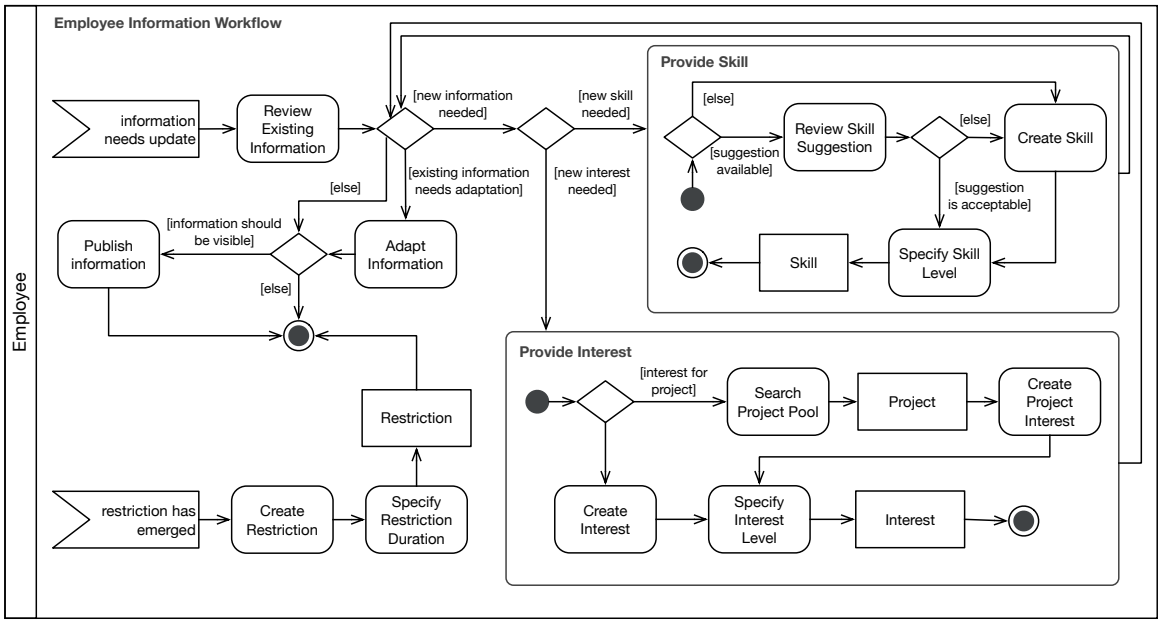


Figure 3.6: Employee Information workflow (UML Activity Diagram)

3.6 Object Design

This section refines the packages of the Analysis Object Model by adding visibilities, method parameters and return types and employing design patterns where necessary [BD09].

In the *Assignment Package*, we introduce a *Strategy* design pattern [Gam95] to decouple the algorithmic generation of suggestions and be able to add, remove, or substitute algorithms transparently. Figure 3.7 shows an excerpt of the Object Design Model of this package, with the newly created objects in color. The `TEMPO Client` uses the `Solver` object, which acts as the *Context* in the Strategy pattern and provides the services to generate suggestions. The available algorithms are described by their abstract `SuggestionAlgorithm` superclass and can be selected at runtime. The `AssignmentProblem` acts as *Policy* by creating a `ConcreteAlgorithm` object and configuring the `Solver` to use it, depending on the `AssignmentCriteria` set up for the problem [BD09].

The refinement of the *Project Package* is shown in Figure 3.8. We introduce the *Composite* design pattern [Gam95] to enable metrics to be composed of other metrics. The `Metric` object is now abstract and can either be a `SingleMetric` or an `AggregateMetric`, each of which can be assigned an individual weight when joined together.

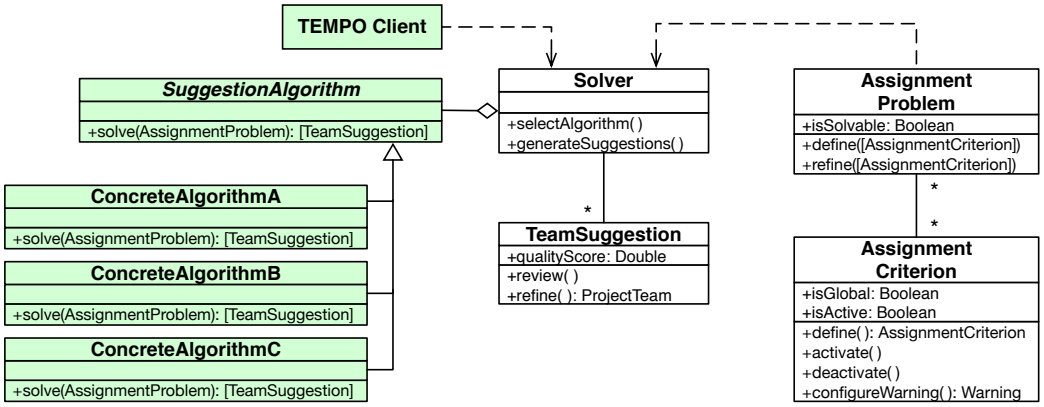


Figure 3.7: Excerpt of the Object Design Model of the Assignment Package (UML Class Diagram)

Moreover, the integration of tools into TEMPO is refined by specifying three concrete types of the abstract `Tool` class, each of which can be queried for different types of data. For instance, while the `Repository` can provide data on the development activity, the `BuildSystem` contains information about builds and releases. Each of these types of `Tool` can be connected to a `Metric`.

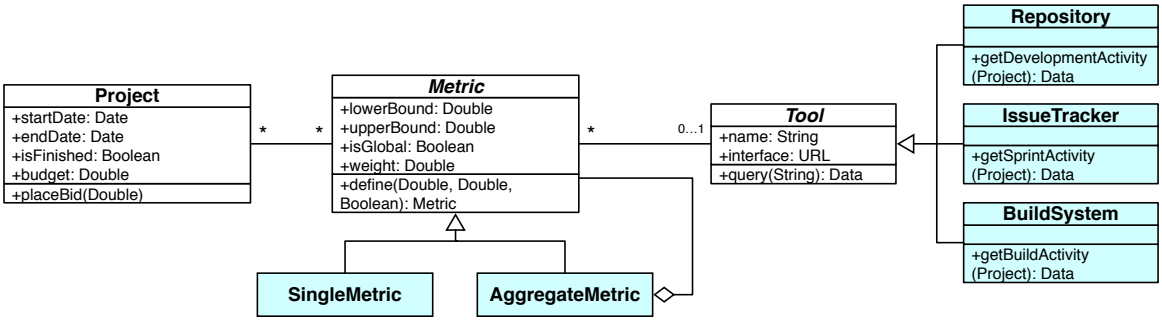


Figure 3.8: Excerpt of the Object Design Model of the Project Package (UML Class Diagram)

Finally, the excerpt of the Object Design Model of the *Employee Package* shown in Figure 3.9 with four concrete types of `Role` to account for different activities performed in a software project. However, TEMPO is not limited to the roles `Developer`, `Tester`, `BusinessAnalyst` and `Architect` and enables the addition or substitution of types of `Role`.

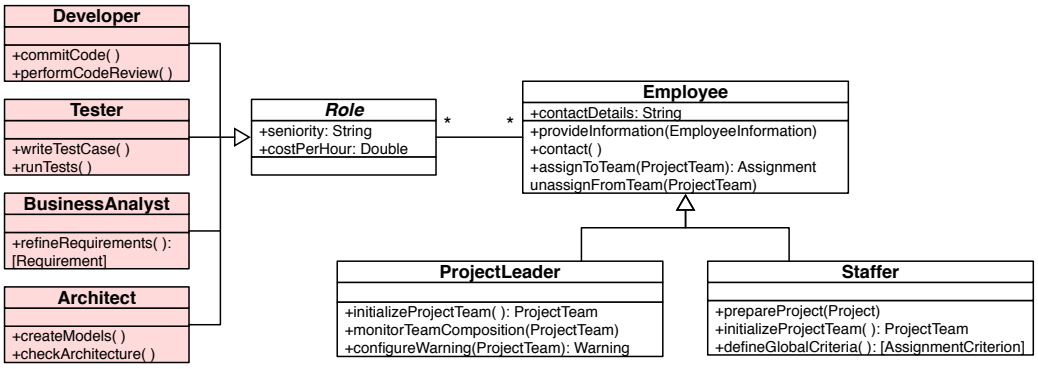


Figure 3.9: Excerpt of the Object Design Model of the Employee Package (UML Class Diagram)

3.7 Suggestion Algorithms

In object design, a Strategy pattern was used to model the algorithmic generation of suggestions in order to enable the dynamic addition or substitution of algorithms (cf. Section 3.6). This section describes multiple algorithmic approaches and discusses whether they can be used within the framework.

In order to be usable for TEMPO's suggestions, a `ConcreteAlgorithm` should be able to solve an `AssignmentProblem` which is defined by a number of `Constraints` and `Objectives`. If the problem is solvable, the algorithm should return a set of `TeamSuggestions` of employees assigned to the team(s) described in the `AssignmentProblem`. The nature of this problem is a classical use case for mathematical optimization [BV04]. Optimization problems have the following form:

$$\begin{aligned} &\text{minimize} && f_0(x) && \\ &\text{subject to} && f_i(x) \leq b_i, && i = 1, ..., m \end{aligned}$$

In the above problem formulation, f_0 is the *objective function*, and $f_i, i = 1, ..., m$ are the *constraint functions*, while $b_i, ..., b_m$ are the *bounds* of the constraints. A solution of this problem is optimal if it has the smallest objective value among all possible solutions. Mathematical optimization can be described as making the best possible choice (meaning the choice with minimum cost or maximum utility) out of a set of candidate choices, while keeping firm requirements in mind that limit this set of possible solutions [BV04].

Optimization problems can be classified by several dimensions: They can be *continuous* or *discrete* depending on the values their variables can take on, they can be *unconstrained* or *constrained* with *linear*, *nonlinear*, or *convex* constraints, and they

can have *one* or *multiple objectives*. Moreover, *deterministic optimization* assumes that the underlying data is accurate, while *stochastic optimization* employs estimation techniques to work under uncertainty [BV04; EW88; LS17].
While the exact nature of the optimization problem is determined by the chosen constraints and objectives, in the cases discussed in this dissertation the team initialization can be expressed as a continuous, constrained, deterministic problem. Section 3.7.1 explores single-objective approaches, while Section 3.7.2 discusses the possibilities of multiobjective approaches. Section 3.7.3 presents other possible kinds of algorithms that can be considered for usage in TEASE in the future.

3.7.1 Single-objective Optimization

A *single-objective optimization problem* has one objective function. If the objective function and constraints are linear, the optimization is in the domain of *linear programming* and well-established solutions such as Dantzig's *Simplex algorithm* exist [Dan51]. These algorithms are quite efficient, which means that one "can easily solve problems with hundreds of variables and thousands of constraints on a small desktop computer, in a matter of seconds" [BV04] and have been used for team assignment problems by fellow researchers [CCFC12] as well as extended to e.g. support fuzzy input data [SG10]. If the problem is nonlinear, it is considerably harder to solve, and algorithmic approaches employ global or local optimization techniques to reach a solution. Therefore, the team assignment problem is expressed in a linear form, first in general terms and then applied to some of the constraints mentioned in the previous chapters.
The criterion for the objective function of the team assignment problem can be selected based on the goal of the organization. For instance, the objective can be to minimize idle time of employees or to maximize their motivation to work on the project. In this general problem formulation, we assume that there is a set of employees $E = \{1, 2, ..., e\}$ being assigned to a set of project teams $T = \{1, 2, ..., t\}$ as well as a variable V_{ij} that expresses the cost of assigning employee i to project j. The basic problem can then be expressed as follows:

$$\text{minimize} \qquad F = \sum_{i \in E} \sum_{j \in T} x_{ij} V_{ij} \tag{3.1}$$

$$\text{subject to} \qquad \sum_{j \in T} x_{ij} = 1 \qquad \forall i \in E, \tag{3.2}$$

$$n_{min} \leq \sum_{i \in E} x_{ij} \leq n_{max} \qquad \forall j \in T, \tag{3.3}$$

$$x_{ij} = \begin{cases} 1 & \text{if employee } i \text{ is assigned to team } j \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

In this problem formulation, an employee can only be assigned to a single team (3.2). Equation (3.3) defines a global team size constraint, where all teams in the assignment problem must have between n_{min} and n_{max} members. A team-based definition of this constraint that applies only to team $t = 1$ would look as follows:

$$n_{min} \leq \sum_{i \in E} x_{i1} \leq n_{max} \tag{3.5}$$

In order to express the remaining criteria, we introduce a generic constraint similar to the approach described in [CCFC12] which allows us to set lower and upper bounds of employees with one particular attribute. The boolean variable $attr_{ik}$ expresses whether employee i possesses the attribute k:

$$attr_{ik} = \begin{cases} 1 & \text{if employee } i \text{ has attribute } k \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

Thus, we can define a set of attributes $A = \{1, 2, ..., a\}$ and set lower and upper bounds for each attribute in the teams:

$$l_k \leq \sum_{i \in E} x_{ij} attr_{ik} \leq u_k \qquad \forall j \in T, \forall k \in A \tag{3.7}$$

The above equation (3.7) can also be modified to refer to a single team, e.g. if the project has special requirements, analogously to setting the team size constraint on a team basis in Equation (3.5).

3.7.2 Multiobjective Optimization

Some argue that a single objective function does not model reality with its conflicting factors well enough, and therefore advocate for *multiobjective optimization problems* with multiple objective functions [LS17]. However, these kinds of problems are considerably more complex to solve: The goal is to find the *Pareto optimal solution*, i.e., the solution from which it is impossible to improve one of the objective functions without deteriorating some of the others [BV04]. Usually multiple Pareto optimal solutions exist, and therefore a mechanism is needed to efficiently find the 'right' one(s). In the following, we discuss some of these mechanisms, which are described in more detail in literature [LS17; Mie12].

One approach to solving a multiobjective problem is *scalarization*, which involves merging the objective functions together, for instance into a weighted sum, and then solving the problem using a single-objective algorithm, thus producing one possible Pareto optimal solution to the multiobjective problem.
In the group of so-called *a priori methods*, a decision-maker specifies information about the relationship between objective functions before the solution algorithm is executed. *Goal Programming* is an approach in this category, in which the decision-maker specifies targets for each objective. The objective is then transformed into the minimization of the sum of the deviations from these targets, enabling a computationally efficient solution of the problem. Another a priori approach is *Lexicographic Ordering*, in which the decision-maker specifies an absolute order of the objectives. The most important objective is optimized first until a solution is reached, after which the second most important objective is optimized with regard to the constraint that the first objective maintains its optimal value, and so on.
The cluster of *a posteriori methods* presents a representation of a set of Pareto optimal solutions to select from to the decision-maker. While the decision-maker does not need to constrain the problem upfront, finding a solution can be computationally expensive, and it is challenging to represent the optimal set in an understandable way when the number of objectives is high. Amongst others, *evolutionary multiobjective optimization methods* can be employed for this approach [CVL02], and they have shown to be popular for obtaining multiple solutions in multiobjective optimization tasks.
Finally, *interactive methods* envision an iterative search process between algorithm and decision-maker. While these approaches are easy to understand for the decision-maker, they also put responsibility on them in finding an optimal solution, which means that they have to have an appropriate understanding of how the algorithm behaves and how to judge the quality of its current result. One example of this category is the *NIMBUS method*, which can handle linear and nonlinear single- and multiobjective problems [MM06]. With this approach, a solution is calculated and presented to the decision-maker, who can then classify the current values of the objective functions into the following categories: (1) should be improved, (2) should be improved until a certain level, (3) is satisfactory at the moment, (4) is allowed to impair until a certain level, and (5) is allowed to change freely. Depending on the decision-maker's choice, a sub-problem is formulated and solved accordingly. The process finishes once the decision-maker is satisfied with the solution. An implementation of the NIMBUS method is available online for academic teaching and research purposes.[2]
In terms of expressing the team assignment problem as a multiobjective optimization problem, we suggest to start from the single-objective problem described in Section 3.7.1

[2] `https://wwwnimbus.it.jyu.fi/`

and to convert some constraints into objectives. For instance, instead of specifying a lower bound for team members with experience in a particular technology, one could formulate an objective to maximize the experience in that technology on the team, possibly up to a certain upper bound. Analogously, one could introduce objectives to minimize idle time of employees while minimizing costs of the project at the same time. These objectives can then either be scalarized or handled using one of the above-described methods depending on the level of knowledge of the decision-maker, their willingness to be involved in the process as well as the nature of the problem: for instance, a lexicographic ordering approach only makes sense if there is an absolute order of importance between objectives, and an a posteriori method is best when a humanly understandable representation of the set of solutions can be achieved.

3.7.3 Further Algorithmic Approaches

While the team assignment problem is an optimization problem by nature, other algorithmic approaches could be considered in this context. For instance, classification and regression methods such as *k-Nearest Neighbors*, *Support Vector Machines*, or *Decision Trees* [FHT01] can be employed to classify teams regarding the quality of their current composition.

Another approach to consider is to use unsupervised learning on historical team composition data to find patterns and thus derive favorable constraints or objectives. For instance, a *Principal Components Analysis* [FHT01] could reveal factors that have been correlated with favorable team performance in the past and could be used as the basis of composing future teams.

For these approaches to work, enough training data on teams with favorable composition needs to be available in the organization. This also means that the organization has to define and document what a 'high-quality team' is, which is not in the scope of this dissertation, as discussed in the following section.

3.8 Prerequisites & Limitations

Like every candidate solution, TEMPO has prerequisites, advantages, and limitations. This section presents conditions under which the framework is designed to be used and discuss its limitations both within and beyond these conditions.

TEMPO is intended to be used in pure project-based organizations, project team organizations and matrix organizations as described in Section 2.1. While it can be useful in other organizational structures, the workflows would likely have to be heavily adapted to fit the organization's processes. Moreover, TEMPO has been designed to be used in organizations with a focus on software engineering, although its approach, for instance the types of roles in the Object Design Model in Section 3.6, can be adapted to fit different contexts.

We predict that the framework is most useful when the organization's Person Pool is large, since this means that there are most probably several options to staff each position. An interview study with practitioners revealed that smaller organizations are struggling more with the challenge of finding *a single* candidate who is available and fits the criteria, than finding *the best* fit for a position [DB18b]. Interviewees also said that they rely most on tools when they do not personally know a suitable candidate, for instance if they are recruiting from another location.

Many of the algorithms described in Section 3.7 do not tolerate missing or ambiguous data. For optimal suggestion results, one should therefore ensure that the *Employee Information* workflow is intact and data about employees and projects is available, or work with algorithms that tolerate fuzzy data.

Finally, an important differentiation to make is that TEMPO's aim is to increase the manageability of the team composition process, not necessarily to create better quality teams. By basing the team initialization on pre-determined criteria, the resulting teams will be less biased, more balanced and more comparable, and the criteria described in Section 2.2.1 are founded on academic research and have shown to positively impact teamwork. However, a team's 'quality' is both a question of the definition used as well as the product of a lot of influencing factors along the way. While the team's composition cannot be considered in isolation as a factor impacting its performance, we recommend basing the process on research-backed criteria, and have designed TEMPO to be extensible and enable changes in the criteria as the organization evolves and discovers what works for them.

Chapter 4

TEASE Reference Implementation

> Change is always tough. Even for those who see themselves as agents of change, the process of starting a new thing can cause times of disorientation, uncertainty and insecurity.
>
> Joyce Meyer

This chapter instantiates TEMPO in the context of the *iPraktikum*, a large multi-project capstone course in software engineering. We present TEASE, the **TE**am **A**llocator for **S**oftware **E**ngineering courses, a reference implementation of TEMPO designed to be used in this context.

Section 4.1 describes the setting of the multi-project course, and Section 4.2 portrays its current manual team initialization process. Following this, Section 4.3 presents the tailoring of TEMPO to this process and instantiates its *Project Preparation* and *Team Initialization* workflows. Section 4.4 describes technological and architectural decisions made during the implementation of TEASE.

4.1 The iPraktikum: A Multi-Project Capstone Course

We instantiate TEMPO in the context of the *iPraktikum*, a large multi-project course with 70-100 participants who develop applications in 10-12 teams, each in collaboration with an industry partner acting as client. The iPraktikum was launched in 2008 and has taken place one to two times per academic year since then. In total, we have conducted 17 instances of the course with 169 projects.[1] Two or three *program managers* are responsible for the teaching methodology and organization of the course. The iPraktikum structure, the educational methods and other aspects of the course have been described in further detail in [ADB16; AJDB17; BKW12; DKA14; DHAB18; DAHB18; KABW14; KDXB18].

Prior to the course, all participants need to go through the *Swift Bootcamp*, a week-long applied introduction to the programming language Swift[2] as well as concepts of iOS development. During this week, students submit 10 pieces of homework to their tutor for correction. The Bootcamp helps the program managers to gain a better understanding of each student's skills during the team initialization by assessing the progress of their homework and gathering their tutors' comments. Moreover, during this week all students fill out a questionnaire asking them to self-report their skills and interests in technologies relevant to the course (cf. Appendix B).

The iPraktikum itself is a semester-long capstone course that starts with a *Kickoff* in which all industry partners present their problem to all participants. Based on the presentations as well as the problem statements provided by the industry partners, the students prioritize all teams. The program managers of the course initialize the project teams on the following day as described in Section 4.2. Depending on the size of the course, each project team consists of six to ten developers, one team coach who is an experienced student having participated in the course as a developer before, and a project leader who is a doctoral candidate.

After the Kickoff, the teams go through *Sprint 0*, a period of two to three weeks in which they reduce uncertainty by verifying requirements with the client, analyzing the problem domain and getting acquainted with the technologies involved in the project. They also iteratively create informal models to represent their understanding of the system to be developed [DKA14]. Sprint 0 can be of varying duration depending on the initial level of definition of the project as well as the technological challenges involved. Following Sprint 0, teams start developing in iterations based on the agile process model *Rugby* [KABW14].

In each instance of the iPraktikum, there are two major events with presentations by

[1] `https://ase.in.tum.de`
[2] `https://swift.org`

all teams: The *Design Review* takes place eight weeks after the Kickoff and features the result of the requirements analysis and system design activities of all teams, and the *Client Acceptance Test* is the last milestone of the course where all teams present the results of their projects.

4.2 Team Initialization Criteria & Manual Process

This section describes the manual team initialization process of the iPraktikum as well as its underlying criteria, which will be the basis of the tailoring of TEMPO and the development of TEASE. The process and criteria have been gradually refined since the start of the iPraktikum in 2008.

Team initialization Criteria

Several types of criteria are taken into account when initializing teams for this project course, as shown in Figure 4.1. The reasoning behind the criteria is described in [DAHB18], including academic research to justify their use.

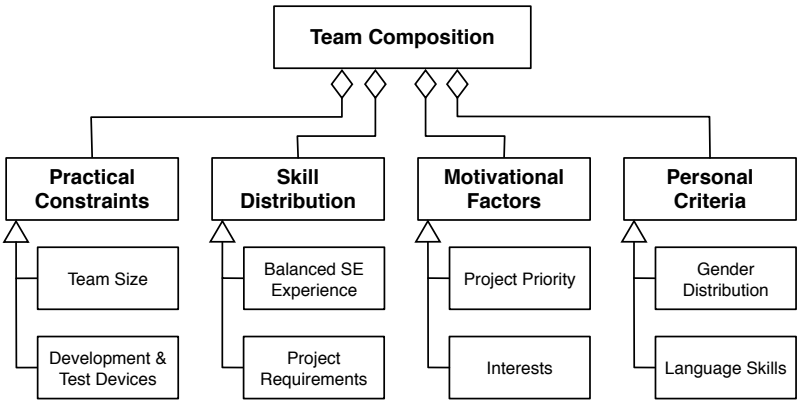


Figure 4.1: Team initialization criteria in the iPraktikum, adapted from [DAHB18]

In terms of *Practical Constraints*, the goal is to create teams of comparable size, except if a particular project needs more or less participants. Moreover, since this course involves development for the iOS platform, we ensure that the number of development devices (e.g. Macs or MacBooks) and test devices (e.g. iPhones or iPads) is above a threshold in all teams. The threshold is pre-set by the program managers based on the average possible number of devices per team and prevents bottlenecks in development and testing in the projects.
When it comes to *Skill Distribution*, the program managers take two criteria into account. Since the iPraktikum is an elective course open to both B.Sc. and M.Sc. level students with multiple majors, participants have a heterogeneous background in

terms of prior experiences and interests, which is why instructors pay special regard to creating balanced teams so that they can learn from each other [DAHB18; KABW14]. Moreover, some projects require skills in a certain technology, in which case the program managers try to find at least one student who is either already skilled in that area, or has stated high interest to learn more about it.
In terms of *Motivational Factors*, the project priority is one of the most important criteria in the team composition process. Our experience shows that students who are assigned to one of their top-priority projects are more motivated to perform well on the team, and we found that this prevents dropouts of students. While the program managers accept a decrease in project priority when it conflicts with other criteria, it is the first factor that is taken into account. The students' self-reported interests in specific technologies are also considered whenever possible.
Finally, *Personal Criteria* such as gender distribution and language skills are taken into account, with the goal of having mixed-gender teams and no isolated foreign students or small, tightly knit groups coming from the same country or community, since this has shown to lead to communication issues on the teams.

Team Initialization Process

The team initialization process is shown in Figure 4.2 as an UML Activity Diagram and illustrated by photos in Figure 4.3. The team initialization is performed on the day after the Kickoff event by the two to three `Program Managers` of the iPraktikum who are educators overseeing the course overall as well as the preparatory Bootcamp (cf. Section 4.1). Students' prior experience is assessed based on self-reported data from the questionnaire they answered before the course starts (cf. Appendix B) as well as information from the Swift Bootcamp. The students' data is printed on `Student Information Sheets`, which are A5-format sheets containing their personal information, interests, project priorities, and their Bootcamp tutor's comments. This information is used to `assign an Instructor Rating` to each student. This rating merges their self-reported information as well as the experience from the Bootcamp into an overall skill level as follows:

1. **Novice (N)**: The student's prior experience in software engineering is low, or they are at the beginning of their studies.

2. **Intermediate (M)**: The student's performance and skills are average.

3. **Advanced (A)**: The student has above-average knowledge in software engineering or has performed remarkably well in previous courses.

4. **Expert (E)**: The student has advanced experience in software engineering as well as with development for the iOS platform (e.g. they have an application in the AppStore).

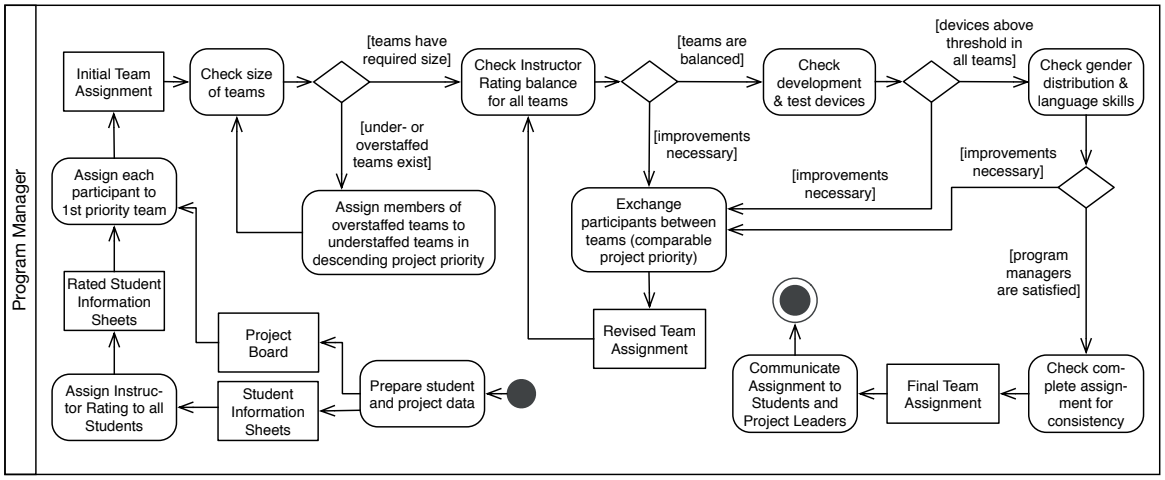
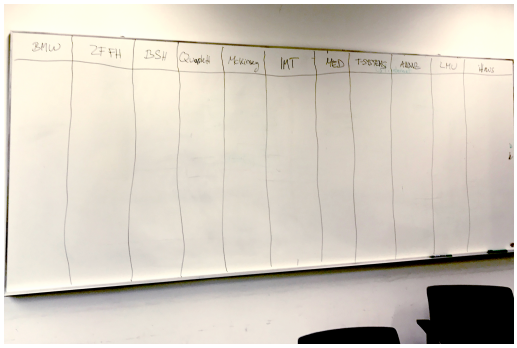


Figure 4.2: Manual team initialization process in the iPraktikum, adapted from [DAHB18]

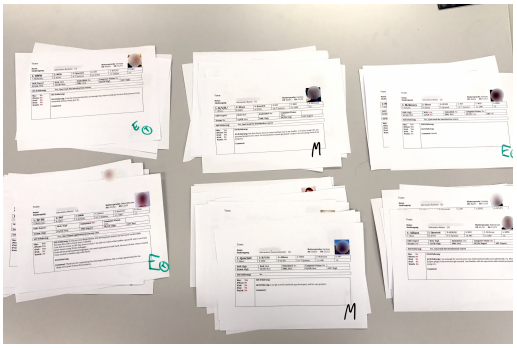
While assigning the Instructor Rating, the program managers take into account their personal experience with each student, as well as cultural factors that might affect their self-assessment. Thus, the Instructor Rating makes students more comparable to each other in terms of prior experience, enabling a better balance of skills on the teams. The result of this step is a set of `Rated Student Information Sheets`, with the Instructor Rating written directly on the sheet in different colors as shown in Figure 4.3b. Moreover, the program managers prepare the `Project Board`, a whiteboard with columns for all projects as shown in Figure 4.3a. If a project has special requirements such as a larger team size or skills in a particular technology, this is written on the board.

After this preparation, the program managers `assign each participant to their 1st priority team` by pinning their sheet in that column on the board, thus creating the `Initial Team Assignment`. Since the teams are usually not equally popular, this results in teams of strongly varying size. Thus, the program managers `assign members of overstaffed teams to understaffed teams`, going lower in students' project priority. The least popular teams are staffed first to ensure their mean priority will not be considerably worse than the average.

Once all teams have the required size, the program managers `check the Instructor Rating balance for all teams`. The goal is to distribute students with an Instructor Rating of *Expert* and *Advanced* equally over all teams, while at the same time not having too many *Novice* students on a team. In order to achieve this, the program managers

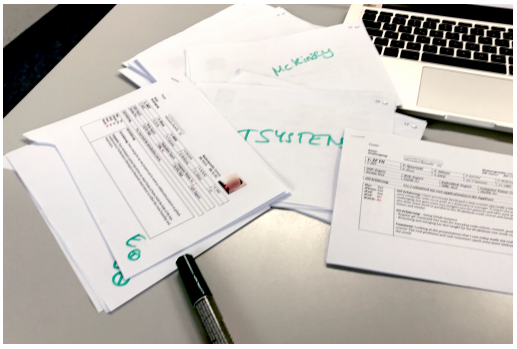


(a) Project Board before initial assignment



(b) Student Information Sheets with Instructor Rating sorted by first project priority

(c) Project Board with Revised Team Assignment



(d) Final Team Assignment (information provided to project leaders)

Figure 4.3: Manual process steps in pictures

`exchange participants between teams` while taking care that their project priority does not decrease considerably. These adaptations create a `Revised Team Assignment` on the board as shown in Figure 4.3c.

As a next step, the program managers `check the development and test devices in all teams`. The devices are manually counted using the information sheets and written next to the project name on the board. If the devices are not above a pre-set threshold, further adjustments are performed. Following this, the same is done for `checking the gender distribution and language skills`: the usual goal is to have at least one female student per team and no single student who does not speak German, since this can lead to communication problems according to our experience [DAHB18].

Each of these checks can trigger further adjustments, in which case a `Revised Team Assignment` is created on the board and all of the previous criteria are checked manually again to ensure that they are still satisfied. This process usually takes between three and five hours. Once the program managers are satisfied with the state of the board, they `check the complete assignment for consistency`, usually by involving a

colleague who has not been part of the process. The `Final Team Assignment` is then taken down from the board and the student information sheets of each finished team, shown in Figure 4.3d, are provided to the project leader so that they can get a first impression of their team. Finally, the program managers `communicate the assignment to the students and project leaders` involved in the course.

4.3 Instantiation of TEMPO's Workflows in TEASE

This section describes the instantiation of TEMPO in the context of the iPraktikum. We tailor the framework to the team initialization process used in the course and show how the process steps are accomplished in TEASE, a reference implementation for TEMPO's *Project Preparation* and *Team Initialization* workflows (cf. Section 3.5). The iPraktikum program managers perform flat staffing of a multi-project organization, i.e., all teams are initialized at the same time with the goal of assigning each member of the Person Pool to exactly one team as described in Visionary Scenario VI.
TEASE covers the entire manual process described in Section 4.2. In order to be able to perform the team initialization, TEMPO's *Employee Information* and *Project Preparation* workflows also need to be instantiated and tailored. In all workflows, the Program Managers take on the role of the Staffer.
The *Employee Information* workflow (cf. Section 3.5.4) is not instantiated since the information is collected only once, right after the Kickoff, and does not need to be kept up-to-date after that. No previous projects exist on the basis of which suggestions can be provided. Instead, students report their skills, interests, and project priorities by selecting from pre-defined categories and levels through the questionnaire in Appendix B. Moreover, the *Team Composition Monitoring* workflow (cf. Section 3.5.3) is not instantiated since there is no need to make changes to the teams' composition during the 3-month-long course.
The following sections describe the instantiation of the *Project Preparation* and *Team Initialization* workflows and show the steps of the process in TEASE's user interface.[3]

[3]The figures of the user interface show the finished UI as refined during the dynamic validation described in Section 5.3, using anonymized or fictitious student and project data.

4.3.1 Project Preparation

The student information is used in the *Project Preparation* workflow shown in Figure 4.4, instantiated based on the original TEMPO workflow described in Section 3.5.1.

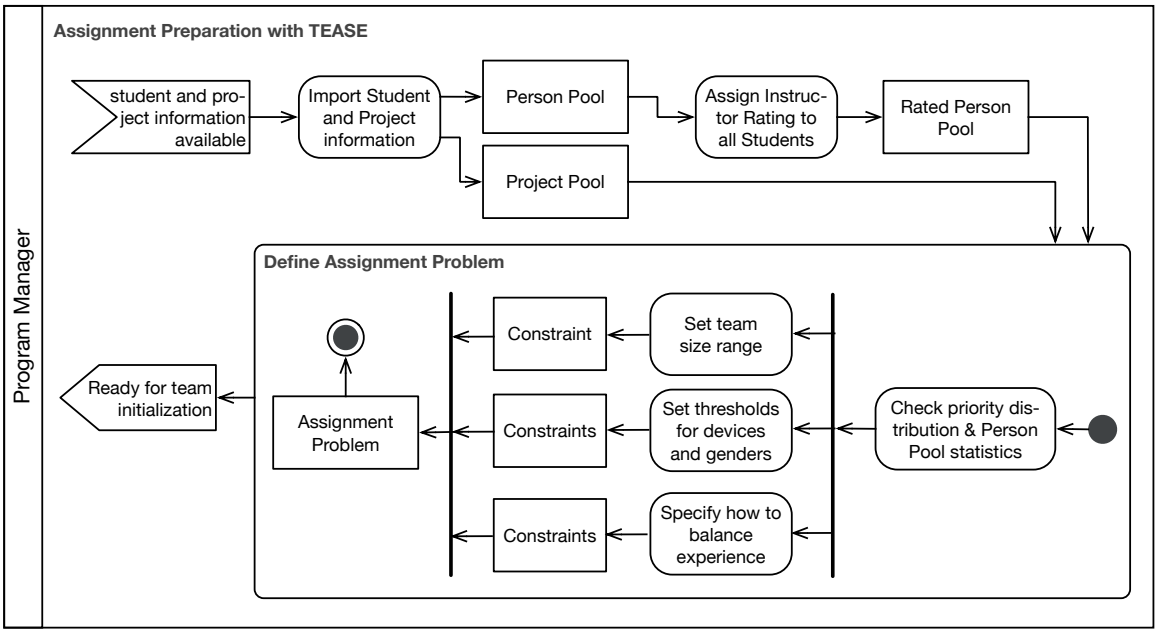


Figure 4.4: Project Preparation workflow in TEASE (UML Activity Diagram)

The `Program Manager` first `imports the Student and Project information`, resulting in the `Person Pool` and `Project Pool`, which replace the `Student Information Sheets` and `Project Board` from the manual process (cf. Figure 4.2). Figure 4.5 shows an excerpt of both pools in TEASE after the import. While the projects look similar to the columns of the empty board, the `Person Pool` consists of an overview of each student in the form of a card.
Just like during manual initialization, the next step is to `Assign an Instructor Rating to all Students` based on their self-reported information. Each student's card can be opened to show more details as depicted in Figure 4.6. Where this data comes from is described in detail in Section 4.4 and [DAHB18]. The following information is shown on this screen:

1. **Personal details:** This section contains identifying personal details of the student, including gender, name, picture, major and degree program, semester number, language skills, development and test devices as well as email address.

2. **Project priorities:** The prioritization of projects the student specified after the client presentations at the Kickoff event (cf. Section 4.1).

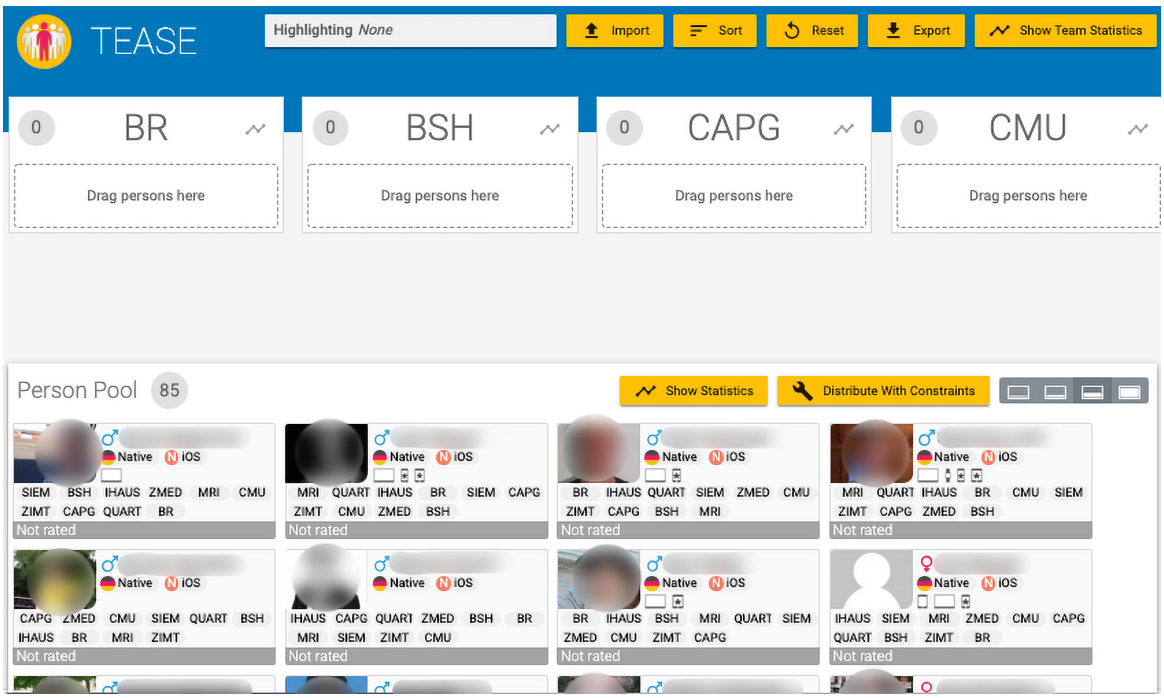


Figure 4.5: TEASE UI: Project Pool and Person Pool

3. **Skills and interests:** We ask for prior knowledge in domains and technologies which are relevant for the iPraktikum. In addition, we also gather interests, assuming that if the student has no experience, but is interested to learn, they will develop their skills during the course. A short explanation is required if the student claims to have above-average experience. Answers are color-coded to enable a quick overview when rating the student.

4. **Experience with student:** Self-reported data is combined with our prior experience with the student in the Bootcamp. We compare the students' perception of the Bootcamp with their tutor's rating and display the tutor's comments to validate the students' self-reported statements before assigning an Instructor Rating.

5. **Instructor Rating:** After reviewing the information, the program manager assigns a rating from *Novice* to *Expert* (cf. Section 4.2). The ratings are used both during algorithmic and manual assignment to create a balance in the teams. Once Instructor Ratings have been assigned to the students, they are shown in a colored label at the bottom of the student's card.

The program managers use this information when assigning Instructor Ratings as well as in the `Review Course Assignment` and `Manually adapt Course Assignment` steps during the *Team Initialization* workflow. In terms of Instructor Ratings, TEASE offers

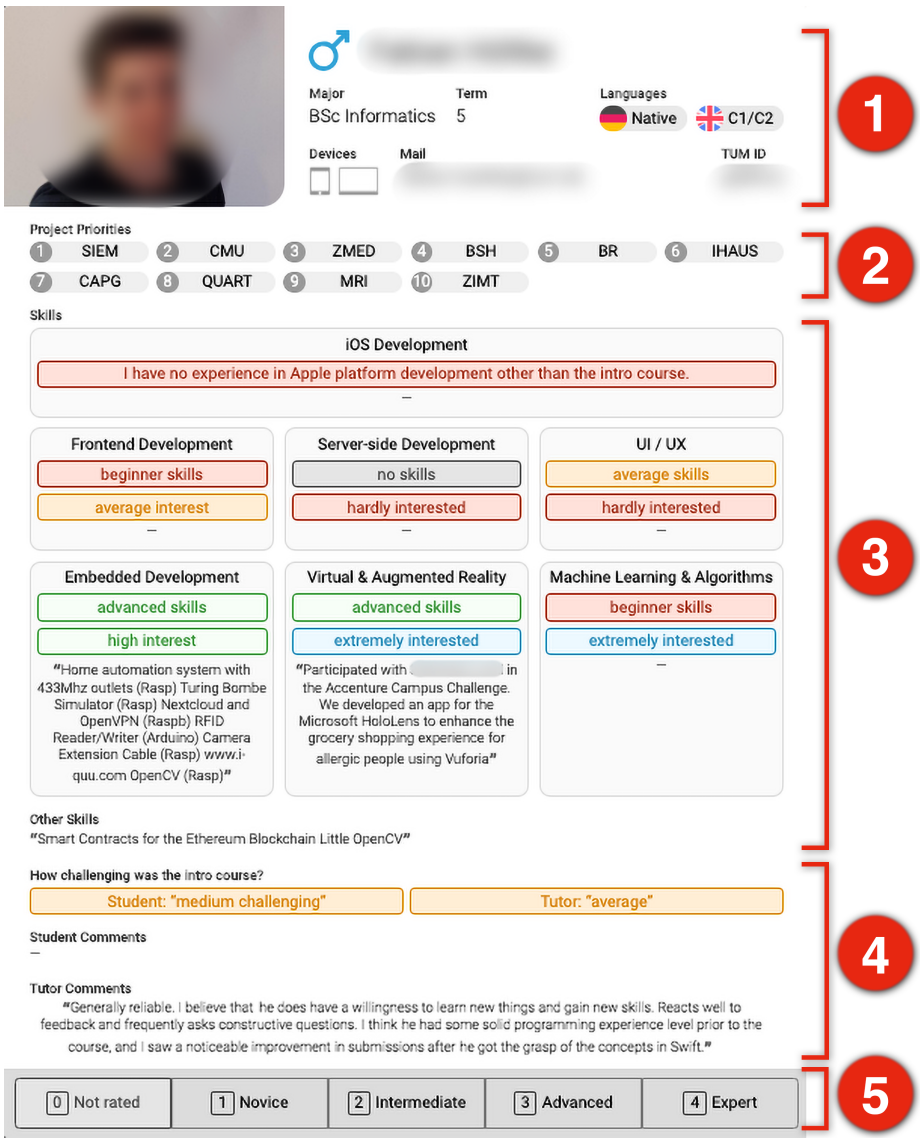


Figure 4.6: TEASE UI: Self-reported and tutor-reported data about a student

the same categories as the program managers use in the manual process, but instead of writing the rating on a sheet, it is available and clearly shown in the tool as part of the `Rated Person Pool`.

Instead of iteratively optimizing for one criterion at a time like in the manual approach, the TEASE-supported tailored process has a `Define Assignment Problem` step (cf. Figure 4.4). For the context of the iPraktikum, we select a single-objective approach that minimizes the project priority (cf. Section 4.4). Thus, apart from the project priority, all other criteria are specified as constraints.

In order to derive the appropriate values for these constraints, the program manager `checks the priority distribution and Person Pool Statistics`. Figure 4.7 shows TEASE's user interface for this step.

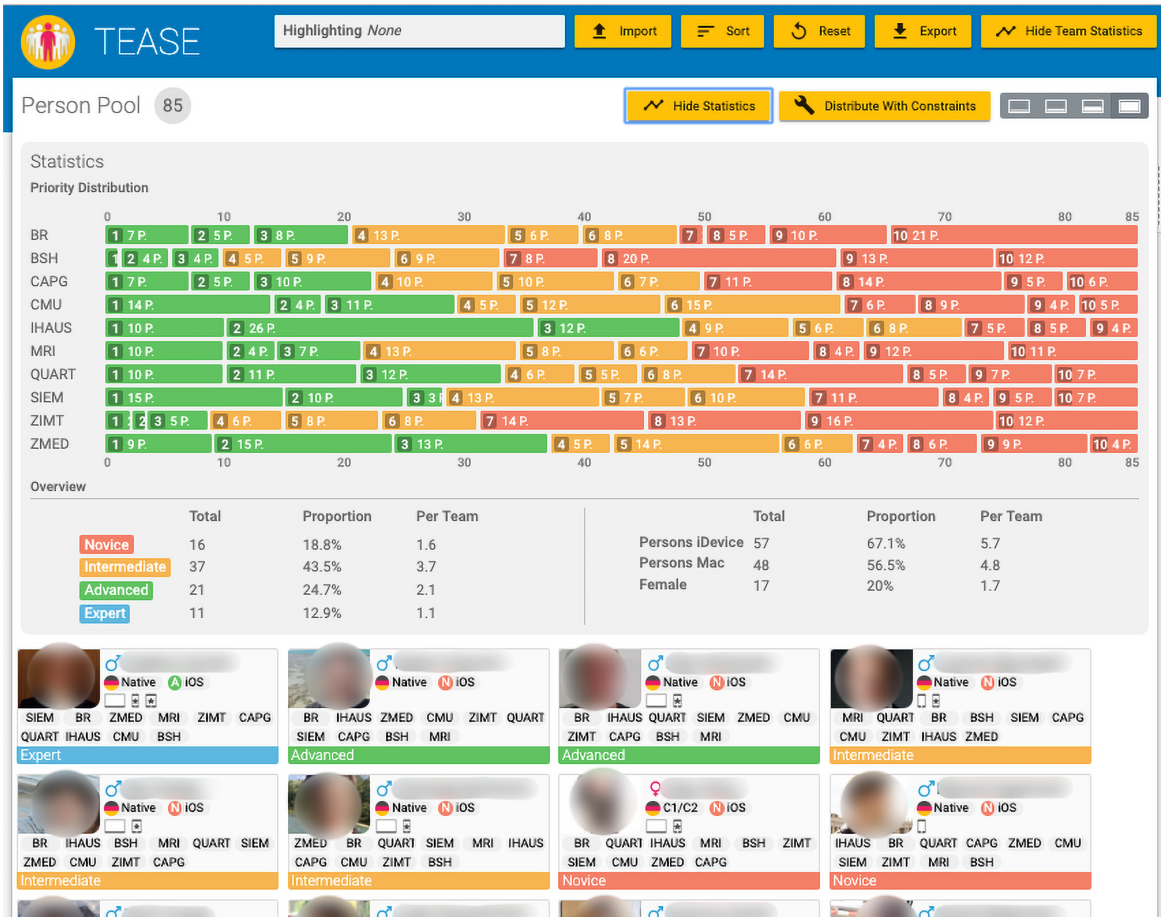


Figure 4.7: TEASE UI: Priority distribution and Person Pool statistics

At the top, this screen contains the `Priority Distribution`, i.e., how all students have prioritized all projects. The program managers can use this overview to determine which projects were less popular than the average, for instance. Below that, the statistics include the distribution of Instructor Ratings, as well as development and test devices and female students. All of these are shown in terms of absolute values in the pool and average values over the current number of teams.

Following this, the program manager `sets the team size range` as well as `thresholds for devices and genders`. With regard to `specifying how to balance experience`, TEASE offers the possibility to create further constraints to distribute students with a particular Instructor Rating equally over the teams. These constraints can be set globally or individually by team. The resulting `Constraints` define the `Assignment Problem`. Figure 4.8 shows how this step is realized in TEASE.

Overall, the tailored *Project Preparation* workflow is less complex than the original TEMPO workflow shown in Figure 3.3, since the educational setting of the iPraktikum does not require budget considerations to be taken into account. Moreover, as all students are developers and there are no distinguished roles on the teams, there is

no need for a `Configure Team Template` step. Thus, the tailoring also simplifies the Object Design Model of the `Employee Package` shown in Figure 3.9 by only using the `Developer` type of the `Role` class.

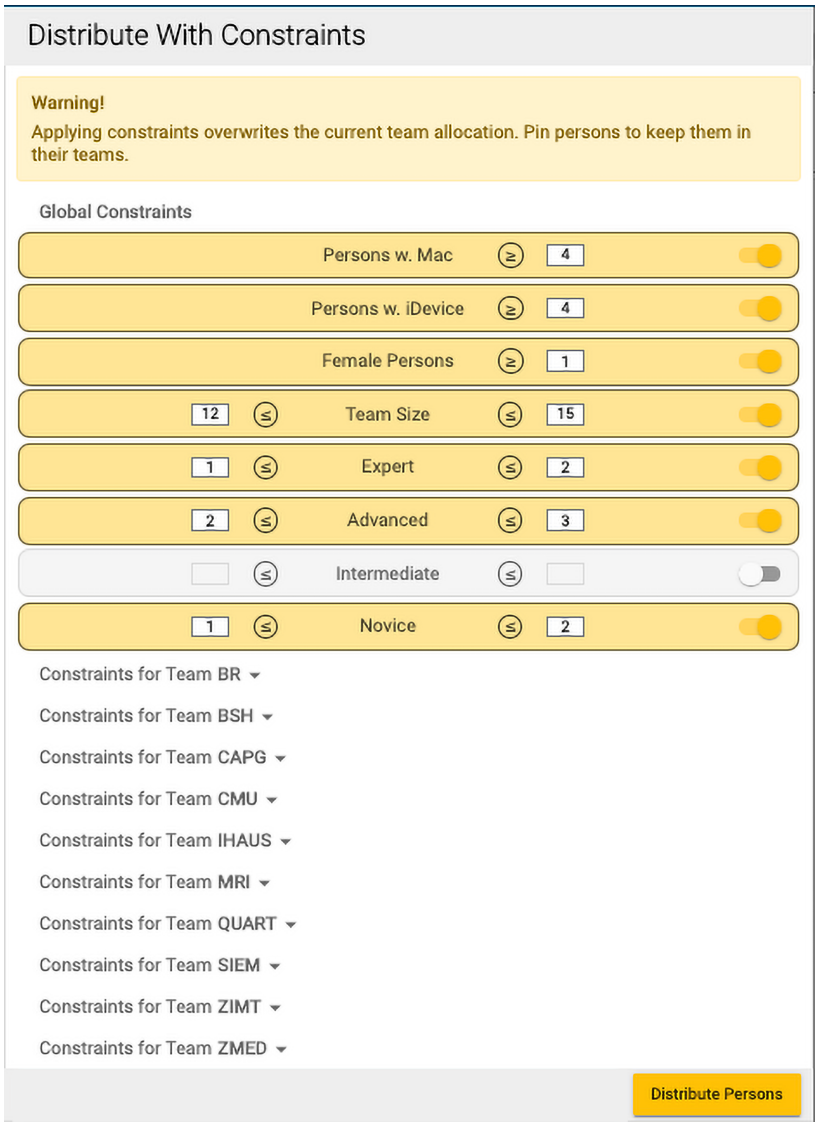


Figure 4.8: TEASE UI: Define Assignment Problem step

4.3.2 Team Initialization

TEMPO's *Team Initialization* workflow is instantiated and tailored as shown in Figure 4.9. The most fundamental adaptation to the original process shown in Figure 3.4 is that instead of initializing a single team, multiple teams are fully staffed at the same time, with each person in the `Person Pool` getting assigned to exactly one team. Thus, instead of the `Team Suggestions` in the original process, TEASE provides algorithmic `Course Suggestions` based on the previously defined criteria. The original `Review Assignment Criteria` and `Adapt Assignment Criteria` steps are also not needed, since in this context the process takes place just after defining the criteria, as opposed to after the organization has won a tender as described in Section 3.5.2.

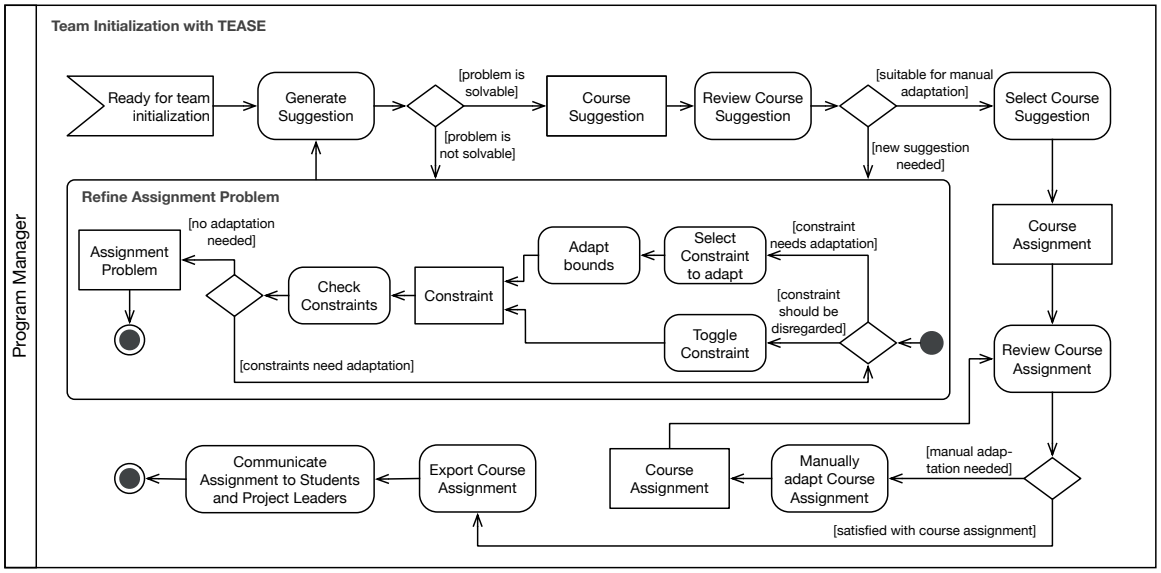


Figure 4.9: Team Initialization workflow instantiated in TEASE

The process starts when the `Program Manager` decides to `generate a suggestion`. Instead of manually optimizing for a single criterion at a time, the reference implementation takes all criteria into account at the same time when generating suggestions. TEASE acts as the `TEMPO Client` described in the Object Design Model in Figure 3.7 and executes the chosen single-objective `Algorithm` with the `Objective` to minimize the project priority and the previously specified `Constraints` (cf. Section 4.4).
If the problem is not solvable, this is due to the previously set criteria. In this case, a `Refine Assignment Problem` step needs to follow, in which the program manager can `select a Constraint to adapt` and `adapt its bounds`, or alternatively `toggle a Constraint` to no longer be taken into account. In TEASE, this step takes place in the same screen where the user defined the first constraints, shown in Figure 4.8. This step is described in greater detail than the `Adapt Assignment Criteria` activity in

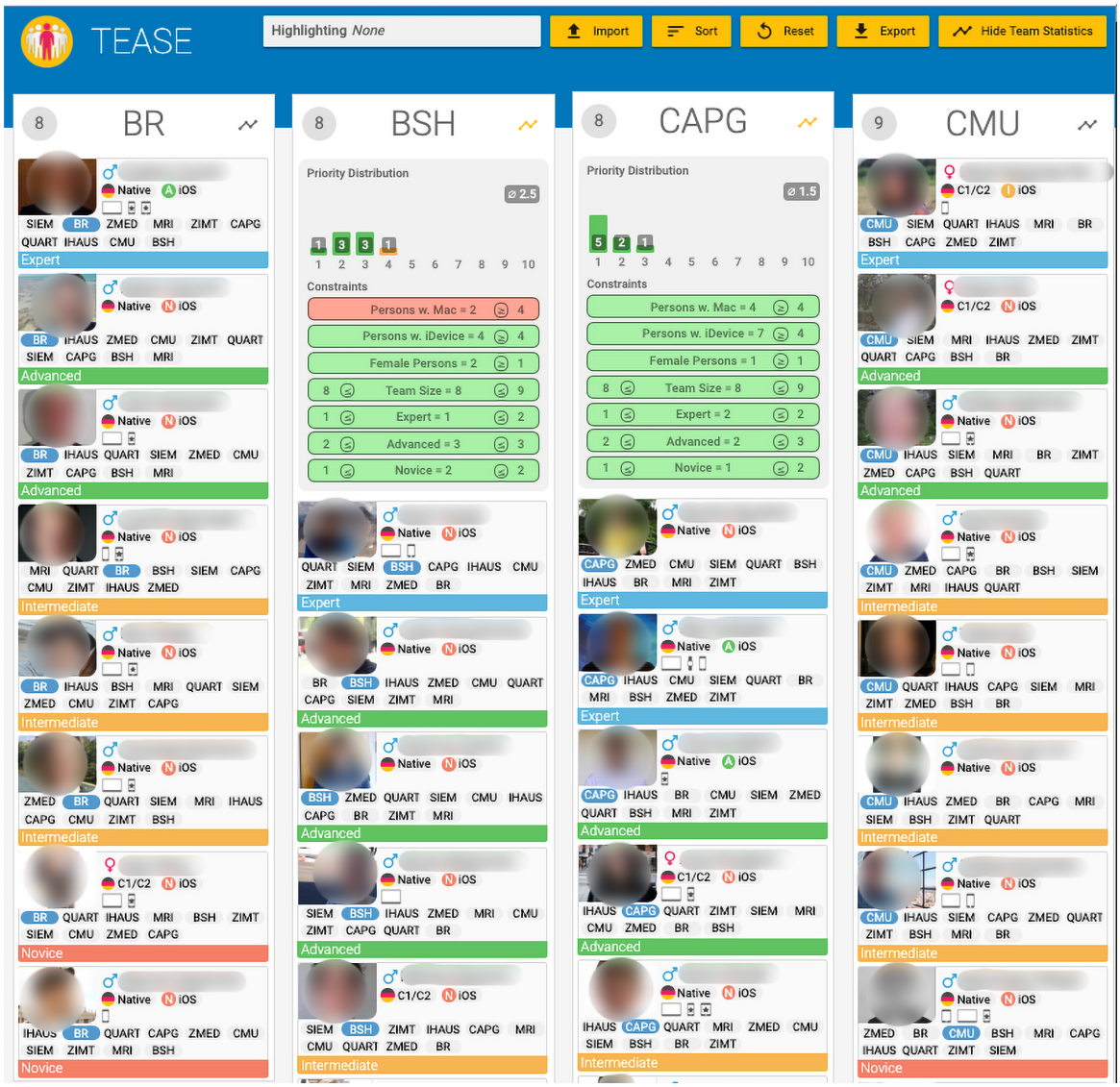


Figure 4.10: TEASE UI: Review Course Assignment step with project statistics

the original process, because in the context of the iPraktikum, the whole `Person Pool` is assigned at once. This makes it more likely that adaptations are needed because no possible solution is found than in the original TEMPO workflow that describes a setting in which a single project team is staffed from a considerably larger pool. The resulting `Constraints` define a new `Assignment Problem`, on the basis of which the program manager can `generate another Suggestion`.

If the problem is solvable, the program manager is presented with a `Course Suggestion` that assigns each student to exactly one project team. This is equivalent to the `Team Suggestions` of TEMPO's original process, except that it contains multiple teams and therefore only a single solution is presented to the program manager. They now `review the Course Suggestion` and can again `refine the Assignment`

`Problem` if they find that it is too far from the intended result. If they find it suitable for manual adaptation, they can `select the Course Suggestion`, resulting in the initial `Course Assignment`.

The `Review Course Suggestion` and `Review Course Assignment` steps look similar in TEASE and are shown in Figure 4.10 as an excerpt of the course with four teams. Each team's column is filled with students assigned to that team and can be sorted by descending Instructor Rating to get a quick overview of the skill balance. TEASE also offers a view of the statistics of the current team composition, which includes the priority distribution of the assigned students, the average priority (i.e., the value of the objective function of that team), as well as an overview over the given constraints and whether they have been fulfilled. The statistics can be opened and closed for all teams or individually per team as shown in the figure.

The program manager can now `manually adapt the Course Assignment` in iterations, which conforms to the `Adapt Suggestion` step of the original process. Adaptations are performed by dragging the cards between teams. With each change, TEASE visualizes the effect on the assignment by adapting the team statistics. In addition, a warning is shown if the program manager's action breaks a constraint.

If the program manager wants to find students with specific attributes, they can use the highlighting functionality shown in Figure 4.11. TEASE allows to add one or multiple criteria to locate students with a minimum skill or interest level in a specific domain, a level of Instructor Rating, a specific gender, or with a particular development or test device. Students who fit these criteria are highlighted in the current assignment so the program managers can quickly find them.

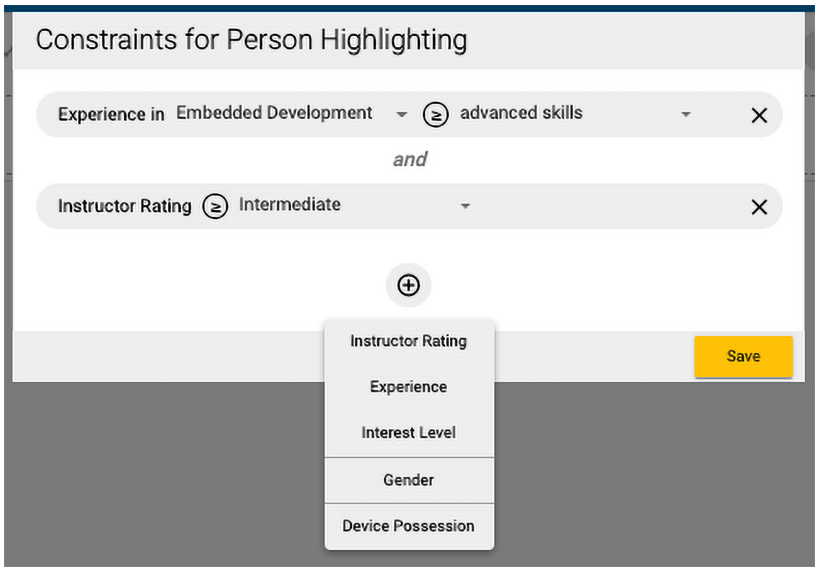


Figure 4.11: TEASE UI: Menu to highlight students

Once the program manager is satisfied, they can `export the Course Assignment`. Since the assignment is available in a digital form, they can immediately `communicate it to the students and project leaders` of the course.

4.4 Implementation & Technological Decisions

This section describes technological, architectural and implementation-specific decisions taken during the development of TEASE.

Architecture & Data Flow

In terms of architecture, we determined that in order to perform the team initialization in the iPraktikum, no persistent data storage is needed, since student data is not saved beyond the initialization. In fact, keeping persistent records of personal data beyond the time frame during which it is absolutely needed for teaching is against the General Data Protection Regulation and the privacy guidelines of the university.[4] The need for data security and privacy was also reflected in the non-functional requirements **NFR3** and **NFR4**. Moreover, we determined that there was no need for distributed access to the Person Pool or the current assignment, since the team initialization takes place with co-located program managers. Thus, we decided not to deploy TEASE to a server, but run it locally on one of the program manager's computers without the data leaving that machine.
Figure 4.12 shows an informal overview of TEASE running on a local machine and the data inputs and outputs of the tool. Import and export of data takes place through comma-separated text files without sending them over the internet and needing to take care of data encryption and security measures. Before the team initialization begins, the program managers import the necessary data into TEASE:

Skills & Interests Questionnaire Data During the Swift Bootcamp before the iPraktikum (cf. Section 4.1), students fill out a questionnaire about their skills and interests in technologies relevant to the course. The full questionnaire is available in Appendix B.

Project Priorities Questionnaire Data At the Kickoff meeting, students watch all client presentations and prioritize the projects (for t projects, this results in descending priorities $1, ..., t$).

Bootcamp Tutor Comments Each student has a dedicated tutor that corrects all of their homework during the Swift Bootcamp (cf. Section 4.1). The tutors shortly

[4] `https://www.datenschutz.tum.de/`

comment on each student and rate their current skill level from their experience during the Bootcamp. This information is used to get an additional perspective on the students' self-assessed skills.

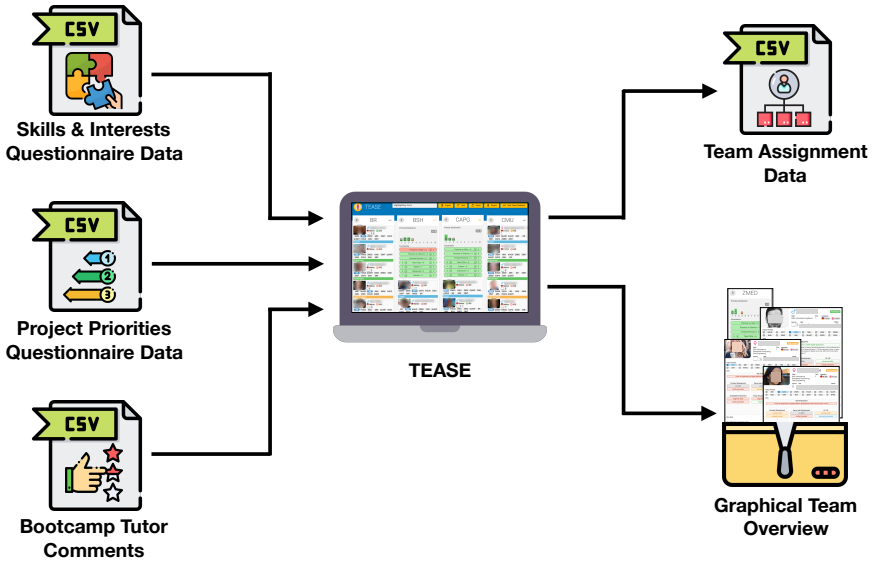


Figure 4.12: Informal overview of data inputs and outputs in TEASE

After the team initialization, the program managers export the results in two formats:

Team Assignment Data The resulting team assignment is exported in a comma-separated format so that the program managers can easily use it for sending serial emails and setting up the team infrastructure.

Graphical Team Overview The teams are exported in a graphical form, including the overview of each team with the resulting objective function value and visualization of satisfied and broken constraints, as well as the student detail cards shown in Figure 4.6. This information is forwarded to the project leaders so that they can get a first impression of who is on their team (cf. Section 4.3).

Frameworks & Technologies

TEASE's user interface was realized with the popular open-source web application framework AngularJS,[5] running on the JavaScript Runtime Node.js.[6] Both were designed with scalability and performance in mind. The tool has been written in TypeScript, an open-source typed version of JavaScript. AngularJS makes heavy use of two patterns: *Model-View-Controller* and *Dependency Injection*.

The *Model-View-Controller* pattern [Gam95] is used to decouple the user interface, described using HTML, from the controller written in JavaScript that executes the application logic and the model containing the data.

Dependency Injection is heavily used in AngularJS applications to increase efficiency, testability and modularity: In each class, the developer can define dependencies using a specific syntax, which are then provided to that class at runtime by the dependency injection framework, thus allowing for greater separation of concerns. Both patterns are used in the implementation of TEASE.

[5] `https://angularjs.org/`
[6] `https://nodejs.org/en/`

Suggestion Algorithm

The algorithmic generation of solutions in TEMPO was described using a Strategy pattern in the Object Design Model Figure 3.7, with TEASE as the `TEMPO Client` that executes the chosen algorithm. For the scope of the reference implementation we selected a single-objective approach as described in Section 3.7.1. The project priority was chosen as the objective based on our experience that assigning students to their top-priority team improves their motivation and prevents dropouts [DAHB18]. Linear programming was used to formulate the problem, and an implementation of the Simplex algorithm [Dan51] to generate solutions, which are also described in detail in [DHAB18].
Starting from the problem formulation in Section 3.7.1, the team assignment problem was expressed in the following way, with a set of students $S = \{1, 2, ..., s\}$ being assigned to a set of project teams $T = \{1, 2, ..., t\}$ and $P_{ij} \in \{1, 2, ..., t\}$ signifying the priority student i gave to project j (lower priority value means they expressed more motivation for the project):

$$\text{minimize} \qquad F = \sum_{i \in S} \sum_{j \in T} x_{ij} P_{ij} \tag{4.1}$$

$$\text{subject to} \qquad \sum_{j \in T} x_{ij} = 1 \qquad \forall i \in S, \tag{4.2}$$

$$n_{min} \leq \sum_{i \in S} x_{ij} \leq n_{max} \qquad \forall j \in T, \tag{4.3}$$

$$x_{ij} = \begin{cases} 1 & \text{if student } i \text{ is assigned to team } j \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Furthermore, the constraints for minimum number of test and development devices, female students and each level of Instructor Rating were expressed using the generic notation for constraints using attributes in **??** in Section 3.7.1.
This approach was selected because of its simplicity and understandability: The program manager is only concerned with setting constraints, while the project priority is optimized automatically. Since the constraints are verifiably true or false, it is easy for the program manager to understand the quality of the suggestion as well as the effects of a manual change on the criteria. While the Simplex algorithm does not deal well with missing data or a very large number of criteria, we predicted that it would be performant and flexible enough for the given setting, which is evaluated in the next chapter.

Chapter 5

Evaluation

> You can't start a product simply by building it. You have to know why you're building it, and you might go down the wrong rabbit hole, waste time, and confuse things. Spending long afternoons with a sketchbook or talking through your ideas with other people can save a year in software development later on.
>
> Mike Krieger

This chapter presents the evaluation of TEASE as an instantiation of TEMPO in the context of the multi-project course *iPraktikum*. We opted for an evaluation in the context of this course because it is close to industry and at the same time open to be empirically studied. Moreover, it is under the control of the researchers to make adaptations based on the findings, providing the possibility to iteratively improve and re-evaluate the solution. The iPraktikum is described in detail in Section 4.1.
We evaluated TEASE in an embedded case study, which is "an empirical enquiry that draws on multiple sources of evidence to investigate one instance [...] of a contemporary software engineering phenomenon within its real-life context". An embedded case study examines multiple units of analysis within a single case – in this approach, the units of analysis are the impact of TEASE on the team initialization process, the improvement of its tailoring to the process as well as its usability and acceptance [RHRR12].

We performed the evaluation steps *lab validation*, *static validation* and *dynamic validation* of the Technology Transfer Model [GGLW06] and extended them based on the recommendation by O'Leary and Richardson [OR12] to validate the solution in multiple stages using different research methods as described in Section 1.2. An overview of the evaluation approach in terms of objectives, research questions, and methods used in each step is shown in Table 5.1. In the following, we give an overview of how these build on each other and provide a bigger picture of the applicability of TEMPO and TEASE to this setting.

The *Lab Validation*, presented in Section 5.1, evaluates the effect of TEASE on the team initialization process of the iPraktikum. We investigated whether the tool is applicable to the context of the multi-project course, and how it affects the process in terms of duration and assignment quality. To achieve this, we conducted a quasi-experiment by performing a TEASE-supported team initialization with the program managers of the course using data from a past iPraktikum instance. We then compared the results to the original, manual initialization which took place at the beginning of that past semester.

Based on the *Goal-Question-Metric Model* (GQM; [BW84]), the following metrics were selected to measure the impact of TEASE on the process: *duration of each process step*, *value of the objective function* (i.e., the mean project priority of students on the team), and *number of broken constraints*. We hypothesized that the usage of TEASE will shorten the duration of all process steps compared to the manual assignment, while achieving a comparable or more optimal level of objective function value and breaking less constraints. In addition to performing a complete TEASE-supported initialization, a subsequent experiment was run by executing TEASE's suggestion algorithm on two other prior instances of the course and analyzing the assignment quality to verify the effect.

Immediately following the lab validation, we performed a *Static Validation* with the same objective in the form of a focus group workshop with the program managers. This evaluation step is described in Section 5.2 and served to enrich and to qualitatively analyze the results of the lab validation. The workshop had both an explanatory as well as exploratory purpose: The subjects reflected on the reasons behind the effects observed in the lab validation and collected feedback on TEASE. The research questions aimed at gaining a better insight into the program managers' experience performing the team initialization with the tool, as well as to elicit requirements that have to be realized before it was ready to be evaluated in the solution domain.

The subsequent *Dynamic Validation I*, reported in Section 5.3, had the goal of improving the usefulness of TEASE in the context of the iPraktikum. The research questions investigated if the tool could be tailored to cover all steps of the, previously manual,

Table 5.1: Evaluation overview: objectives, research questions, methods

Evaluation step	**Objective**	**Research Questions**	**Method(s)**
Lab Validation (Section 5.1)	Evaluate the applicability and effect of TEASE on the team initialization process in the context of the multi-project course.	**RQ1** Is TEASE applicable to the context of the multi-project course? **RQ2** How does TEASE affect the process in terms of duration and assignment quality?	Quasi-experiment
Static Validation (Section 5.2)		**RQ3** What is the program managers' experience performing the team initialization with TEASE? **RQ4** Are there requirements that need to be fulfilled before the tool can be released?	Focus group workshop
Dynamic Validation I (Section 5.3)	Improve the usefulness of TEASE from the perspective of the program managers of the multi-project course.	**RQ5** Can TEASE be tailored to cover all activities and requirements of the existing team initialization process? **RQ6** Which new requirements emerge beyond the original process?	Action Research, Focus group workshop (3 iterations)
Dynamic Validation II (Section 5.4)	Evaluate the usability of and attitude toward TEASE from the point of view of domain experts who have never used the tool before.	**RQ7** Can new users successfully perform the team initialization with minimal prior training? **RQ8** What issues do the participants have when using the tool and how challenging do they find the tasks? **RQ9** Are the participants satisfied with TEASE as a solution for performing the team initialization?	Observational study, Questionnaire

team initialization process, as well as which new requirements would emerge that go beyond the original process, but further help the program managers perform the team initialization. In order to answer these questions, the program managers performed the complete team initialization in three consecutive instances of the iPraktikum.

Instead of a case study which is purely observational, we opted for action research, which is "focused on and involved in the change process" [RH09a]. After each run, we conducted a new focus group workshop to collect feedback and requirements from the program managers. The elicited requirements were labeled as either process gaps or new requirements, prioritized and used to iteratively improve the tool before the next run.

Following the third iteration, we conducted a **Dynamic Validation II** step in the form of an observational study using the think-aloud protocol with the project leaders of the course, presented in Section 5.4. The study subjects were domain experts: They had led multiple projects and therefore knew what criteria affected the team's performance and morale. However, they had never used TEASE before. The objective was to evaluate the usefulness and attitude toward TEASE using observation, which is said to "provide a deep understanding of the phenomenon that is studied" [RHRR12]. After a short introduction into the context and an overview of the tool, the subjects received a series of tasks to perform using TEASE while thinking aloud [VBS94], followed by a questionnaire. We analyzed the resulting assignment problem configuration in terms of the constraints they set as well as the number and kinds of issues that occurred during the execution of the tasks. Moreover, we examined how challenging they found the tasks as well as their perceived usefulness and ease of use of the tool as defined by the *Technology Acceptance Model* (TAM; [Dav89]) as influencing factors for technology usage and acceptance. Participants also provided open-ended responses regarding advantages and disadvantages of TEASE from their perspective, and indicated whether they trusted the tool and would recommend it to fellow educators.

We present the method, results and implications of each evaluation step in the respective sections, but discuss the limitations and threats to validity for all steps at once in Section 5.5, which also concludes this chapter.

5.1 Lab Validation: Quasi-Experiment

As suggested by the Technology Transfer Model and described in Section 1.2, we first validated TEASE in a controlled setting. The goal of the lab validation was to evaluate the applicability and effect of TEASE on the team initialization process in the context of the iPraktikum. To reach this goal, we studied the following research questions:

RQ1 Is TEASE applicable to the context of the multi-project course?

RQ2 How does TEASE affect the process in terms of duration and assignment quality?

We studied **RQ1** by attempting to perform the team initialization using TEASE, determining whether the program managers can successfully produce an assignment using the tool.
For **RQ2**, we compared the quality of this TEASE-supported assignment to the previously done manual assignment of the same instance of the iPraktikum. In order to compare the two, we needed to define how to measure assignment quality. We derived metrics for assignment quality based on the Goal-Question-Metric Model [BW84; FB14; RH09b; Woh+12]. The GQM is recommended to provide clarity regarding the intended results of the research and to avoid collecting unnecessary data.
From the research goal described in Table 5.1, the two questions *'Does TEASE affect the duration of the process?'* and *'Does TEASE affect the quality of the assignment?'* were derived as shown in Figure 5.1. The metrics *total duration of team initialization* and *duration of each process step* were selected to answer the first question, while the process quality was measured by the *objective function value of the assignment* as well as by the *number of broken constraints*.

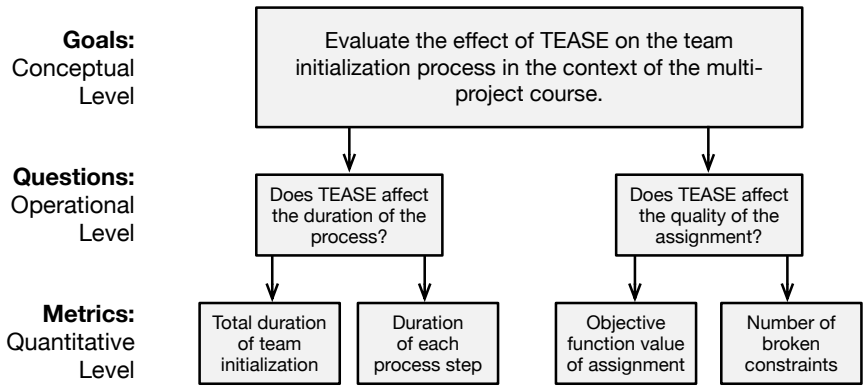


Figure 5.1: Goal-Question-Metric Model for **RQ2**

In the following, we present the experimental design including hypotheses, give details on the execution of the initialization and describe the data analysis procedures. After

a summary of the results, we discuss the findings with regard to the research questions and hypotheses and highlight implications for subsequent evaluation. Threats to validity are discussed in Section 5.5 together with the other validation steps.

Experimental Design

In order to assess its effect on the team initialization process in the context of the iPraktikum, we conducted a quasi-experiment comparing the output of TEASE in its initial Version 0.1 to the previous manual process based on data of the same instance of the course. Figure 5.2 shows an overview over the experimental approach.
Using student information from the iPraktikum of the winter semester 2016/17, three program managers collaboratively performed TEMPO's complete team initialization workflow as tailored to this setting (cf. Section 4.3) and recorded the duration of each process step. The author of this dissertation was one of the program managers performing the process. The resulting team assignment was exported for comparison to the previously done manual assignment. The execution of the team initialization with TEASE was intended to provide an answer to **RQ1** as well as usage data to investigate **RQ2**.

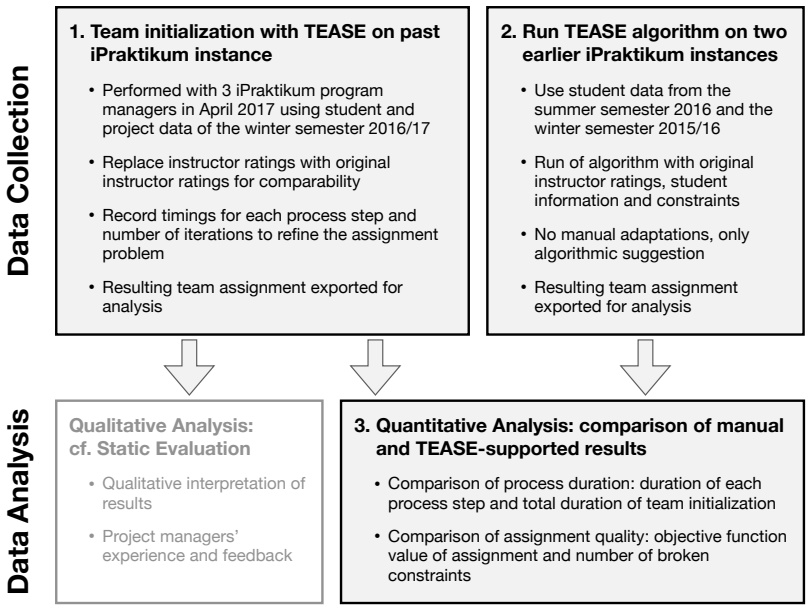


Figure 5.2: Experimental design of lab validation

In order to further investigate **RQ2**, we ran TEASE's suggestion algorithm on the two prior instances of the iPraktikum: summer semester 2016 and winter semester 2015/16. The original Instructor Ratings, student information and criteria were used to generate a solution but did not make manual adaptations or record timings, since these courses were too far in the past to be able to realistically assess project

requirements on participants' skills. Moreover, the student information for these semesters was considerably more limited since the amount of details collected from the course participants was restructured and increased in the winter semester 2016/17. Thus, the objective of this step was simply to determine whether TEASE would have created a more optimal solution for these teams than the manual process, using the same objective and criteria to initialize the teams.
With the results of the TEASE-supported assignment in these 1+2 course instances, the independent variables *duration of each process step* as well as the *total duration of the team initialization* were compared between the two process types. Moreover, the results in terms of assignment quality as measured by the selected independent variables *objective function value of assignment* and *number of broken constraints* were analyzed. The qualitative analysis of the team initialization followed in the static validation step described in Section 5.2.
The following hypotheses were investigated in the quasi-experiment:

H1.1 The program managers are able to produce an assignment for the course using TEASE (**RQ1**).

H2.1 With TEASE, the duration $d_T(s)$ of each process step $s \in S$ is shorter than the manual duration $d_M(s)$ of the same step (**RQ2**). $\forall s \in S : d_T(s) < d_M(s)$.

H2.2 With TEASE, the overall duration of the team initialization process is shorter than the manual execution (**RQ2**). $\sum_{s \in S} d_T(s) < \sum_{s \in S} d_M(s)$.

H2.3 The assignment created with TEASE has a more optimal or equal[1] objective function value F compared to the manual assignment (**RQ2**). $F_T \leq F_M$.

H2.4 The number of broken constraints $|\overline{C}|$ of the TEASE-supported assignment is smaller than or equal to that of the manual assignment (**RQ2**). $|\overline{C}_T| \leq |\overline{C}_M|$

[1]See Section 4.4 for details on the calculation of the objective function value.

Data Collection

In the following, the concrete execution of the team initialization with TEASE on the winter semester 2016/17 data of the iPraktikum is described. Figure 5.3 shows the instantiation of the TEASE-supported process first presented in Section 4.3, including steps from the *Assignment Preparation* and *Team Initialization* workflows.

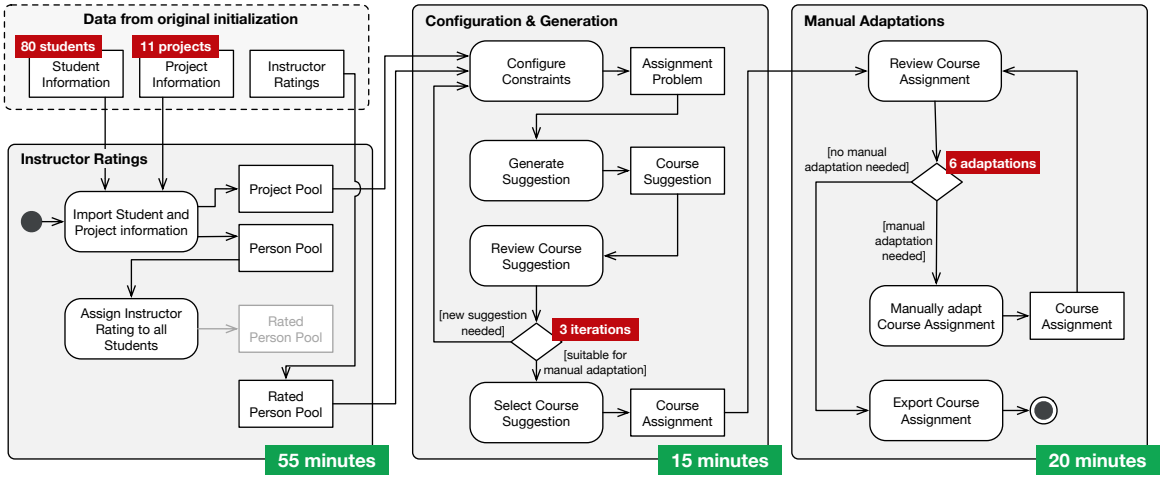


Figure 5.3: Team initialization process as instantiated during the experiment

As shown in Figure 5.3, the program managers were provided with student and project information from the original instance of the course and were instructed to `assign Instructor Ratings to all students`. After this step, the `Rated Person Pool` was replaced using the original Instructor Ratings from the manual initialization. This enabled us to record the duration of this step while at the same time ensuring comparability between manual and TEASE-supported process. We chose not to analyze the differences and similarities between the two sets of Instructor Ratings, since it is very likely that the program managers' perception of the students had been strongly shaped by the experience they had had with them during the course.
For the `Configuration & Generation` step, which is a combination of the original `Configure Assignment Problem`, `Generate Suggestion`, and `Refine Assignment Problem` steps described in Figure 4.9, the program managers `configured constraints` based on the average numbers in the Person Pool as well as the original goals of the manual assignment. After setting initial constraints, they generated a suggestion and reviewed it, finding that the configuration needs adaptation. They refined the constraints until they were satisfied with the initial suggested solution, which took three iterations. The total values of the criteria in the course as well as averages over the 11 teams and the resulting final constraints are shown in Table 5.2. The team size was set to 7-8 and the number of female students to at least 1 per team because this had been the aim of the previous manual assignment. The development

Table 5.2: Person Pool data and constraints set during the experiment

	Total	**Avg. per Team**	**Lower bound**	**Upper bound**
Number of Students	80	7.3	7	8
Instructor Rating: Expert	2	0.2	0	1
Instructor Rating: Advanced	24	2.2	2	3
Instructor Rating: Intermediate	34	3.1	-	-
Instructor Rating: Novice	20	1.8	1	3
Development Devices	53	4.8	4	-
Test Devices	46	4.2	4	-
Female Students	12	1.1	1	-

and test device constraints were set to the maximum possible per team considering the number of devices in the overall Person Pool. Finally, the program managers set constraints for three out of the four Instructor Ratings, namely *Expert*, *Advanced*, and *Novice*. This was based on their experience that the least and most experienced team members should be equally distributed across teams to create a strong starting point, while team members with average skill level could be used to 'fill up' the remaining space on teams or to satisfy constraints.

As soon as the program managers were satisfied with the resulting Course Assignment, they proceeded to the `Manual Adaptations` step. They made a total of six manual adaptations when they judged that the balance of the suggested teams in terms of experience was not optimal (cf. Section 4.3). The process concluded when they did not see the need for any more manual adaptations, at which point they exported the assignment to be compared to the manual process, the results of which are presented and discussed in the next sections.

Data Analysis

The durations $d_T(s)$ for each process step were recorded and compared to the previously recorded manual durations $d_M(s)$. Moreover, the durations were summed up to create the overall duration of the team initialization process, both for the manual run $\sum_{s \in S} d_M(s)$ and the TEASE-supported run $\sum_{s \in S} d_T(s)$. Table 5.3 shows the results for this step.

In order to determine the assignment quality, the objective function value was calculated for each team $t \in T$: $F_T(t)$ for TEASE-supported and $F_M(t)$ for manual (cf. Table 5.4). The overall objective function values F_T and F_M were calculated by summing up $F_T(t)$ and $F_M(t)$ over all teams in each individual instance of the course. The number of broken constraints $|\overline{C}|$ was also summed up over each instance. The results are summarized in Table 5.5.

Lastly, a Wilcoxon signed-rank test was performed to compare the manual and TEASE-supported objective function values per team. The test was performed individually for each of the three evaluated iPraktikum instances to compare the two related samples with regard to whether the difference in their means is coincidental.

Results: Applicability of TEASE to this Context (RQ1)

The program managers were able to initialize the course using TEASE, producing an assignment in which each of the 80 students from the sample belongs to exactly one of the 11 project teams. This provides support for **H1.1** and the corresponding research question **RQ1**, although further evaluation is needed in a less controlled setting to confirm the findings.

The qualitative analysis of the program managers' feedback regarding the applicability of TEASE took place during the subsequent focus group workshop and is presented in Section 5.2.

Results: Process Duration (RQ2)

In the following, we present the results regarding the hypotheses **H2.1** and **H2.2**. The recorded duration of each process step as well as the duration of the previously done manual initialization are shown in Table 5.3. The usage of TEASE reduced the duration of the `Assign Instructor Rating` step by four minutes, but a larger effect can be observed in the `Configure Assignment Problem` and `Adapt Suggestion` steps. This part of the process took 173 minutes during the manual run and only 35 minutes using TEASE. In total, the process was reduced from 3.9 hours to 1.5 hours, which is a decrease of 61%.

Table 5.3: Results for hypotheses **H2.1** and **H2.2**

Process Step $s \in S$	$d_M(s)$	$d_T(s)$
Instructor Ratings	59 minutes	55 minutes
Configuration & Generation	173 minutes	15 minutes
Manual Adaptations		20 minutes
Total duration	**232 minutes**	**90 minutes**

Since both the duration of the individual steps as well as the total duration were shorter in the TEASE-supported initialization than during the manual run, the hypotheses **H2.1** and **H2.2** have proven to be true in this experiment.

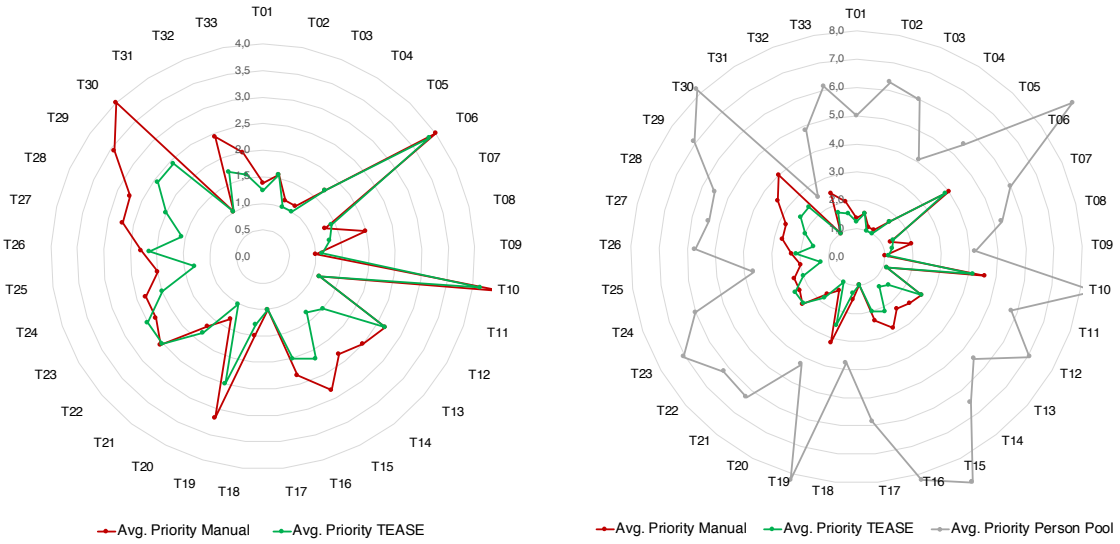
Results: Assignment Quality (RQ2)

The impact of TEASE on the assignment quality was measured by the value of the objective function, which in turn was determined by the average priority of students on each team as well as the number of broken constraints in the overall assignment. The comparison of these variables between manual and TEASE-supported process is shown in Table 5.4. The cases in which the usage of TEASE led to a slight improvement are highlighted in light green, and significant improvements (-0.5 in average priority or higher, or two or more broken constraints less) are shown in dark green. Cases where TEASE led to a deterioration are shown in orange.

Table 5.4: Assignment quality, manual vs. TEASE-supported process

Team (anonymized)		**T01**	**T02**	**T03**	**T04**	**T05**	**T06**	**T07**	**T08**	**T09**	**T10**	**T11**
Avg. priority in course		5.0	6.3	6.0	4.1	5.5	9.4	6.0	5.3	4.2	8.5	5.8
Avg. priority in team	**Manual**	1.4	1.6	1.1	1.1	1.7	4.0	1.3	2.0	1.0	4.6	1.1
	TEASE	1.3	1.6	1.0	1.0	1.7	3.9	1.4	1.3	1.1	4.1	1.1
Broken constraints	**Manual**	1	1	0	0	0	0	1	0	0	2	0
	TEASE	0	0	0	0	0	0	0	0	0	0	0

The results show that TEASE led to an improvement of the mean project priority in more than 50% of the teams, two cases of which (T08 and T10) were significant improvements of more than 0.5. The usage of TEASE led to a deterioration of the objective function by a value of 0.1 in two cases, T07 and T09.
The previously created manual assignment was measured against the same constraints to see how the manual assignment compares to the TEASE-supported assignment. The results show five cases of broken constraints in the manual assignment. Using the tool, all constraints were satisfied even when manually adapting the suggested assignment, since TEASE warned the program managers when breaking a constraint.



(a) Assignment only

(b) Including avg. priority overall

Figure 5.4: Comparison of manual and TEASE-supported team initialization over three semesters in reversed chronological order: T01-T11 winter semester 2016/17, T12-T23 summer semester 2016, T24-33 winter semester 2015/16

The run of TEASE on the two prior instances of the course resulted in a comparison of manual vs. TEASE-supported process in 22 additional teams, bringing the total to 33 teams. An overview of the results is shown in Figure 5.4, both as a comparison of the value of the objective function, i.e. the average priority of students on each team, as well as the same comparison with the average priority over the whole Person Pool added in for reference. The data tables as well as the full-size images and a box plot of the objective function values per team can be found in Appendix C.

Of the 33 teams, 24 showed an improved objective function value in the TEASE-supported process, and in 11 of these cases the improvement was 0.5 or larger. The value of the objective function deteriorated for four teams in total. While manual process had broken 16 constraints over these three semesters, TEASE found a solution without breaking any constraints.

The paired Wilcoxon signed-rank test yielded a significant result for the winter 2015/16 instance ($p = 0.0092$) and the 2016 instance ($p = 0.025$) of the course. This suggests that the mean objective function value of the manually initialized teams is different from the mean of the teams initialized using TEASE. The 2016/17 instance did not yield a significant result ($p = 0.073$), which can be due to the small number of teams in the sample. The box plot of the objective function values per team in Appendix C visualizes these distributions graphically.

With regard to the hypotheses **H2.3** and **H2.4**, Table 5.5 shows an overview over the objective function value of the assignment over all teams in the course $F = \sum_{t \in T} F(t)$ as

Table 5.5: Results for hypotheses **H2.3** and **H2.3**

	F_M	F_T	$\lvert\overline{C}_M\rvert$	$\lvert\overline{C}_T\rvert$
Winter 2015/16	24.94	18.14	4	0
Summer 2016	26.21	22.29	7	0
Winter 2016/17	20.94	19.51	5	0

well as the number of broken constraints $|\overline{C}|$. Since the TEASE-supported assignment yielded a more optimal objective function value in all three instances while not breaking any constraints, we find support for both **H2.3** and **H2.4**.

Discussion & Implications

In the following, we discuss and reflect upon the results with regard to the research questions and derive implications for the subsequent evaluation of TEASE.
Since the program managers were able to produce an assignment for the course, we can provide a positive answer to **RQ1**. However, deeper investigation into the experience of the program managers with the tool is needed to qualitatively understand and support the results. The focus group workshop reported in Section 5.2 was aimed at examining research questions that look more closely at user feedback and eliciting requirements.
In terms of **RQ2**, we investigated how TEASE impacts the process in terms of process duration and quality. When it comes to the duration of the individual process steps, the introduction of TEASE had the largest effect on the assignment itself, while the impact on the `Instructor Ratings` step was negligible. This is most probably due to the fact that TEASE does not change the way this process step is carried out; the tool digitizes the information, shows it in one place and adds the Instructor Rating to the existing data, but is not aimed at speeding up the process of rating itself. The use of TEASE during the subsequent assignment steps showed a reduction of almost 80% in duration. This effect can be attributed to the strength of optimization algorithms to quickly find a solution while taking multiple criteria into account, which quickly becomes difficult to handle for humans as the amount of criteria, and thus also the amount of possible tradeoffs between them, grows. Thus, we found support **H2.1** with the restriction that the effect on the `Instructor Ratings` step is negligible and can be due to other factors. The results also support **H2.2**, since the overall duration of the TEASE-supported process was shorter than the manual initialization.
With regard to the quality of the resulting assignment, TEASE led to an improvement of the objective function, i.e., the average priority of students on the team, in six out

of 11 teams, two cases of which showed a substantial improvement of 0.5 or more. This effect also proved true in the other two instances of the course, with a substantial improvement in one third of all teams and a deterioration in only four cases overall, all of which were smaller than 0.2. The objective function value over all teams in the course was lower than in the manual process in all three evaluated instances, supporting **H2.3**. From the collected data, we cannot draw conclusions as to whether TEASE finds better solutions for teams with a low overall priority in the course, since the largest improvements are spread across teams with varying overall priority. Thus, we asked the program managers in the following lab validation where they assumed TEASE had helped them most.
TEASE found these favorable solutions without breaking constraints in any of the 11 teams, and the program managers did not break any constraints manually either, being alerted by the tool when a change had a negative effect on the assignment's quality. This result supports the hypothesis **H2.4**. In the two earlier semesters, the manual adaptations were omitted from the process and thus cannot say whether the program managers would have broken constraints in this step.
Taking these two measures for assignment quality into account, we can answer **RQ2** by concluding that TEASE considerably shortens the team composition process and leads to an improvement in the overall objective function value of the assignment while reducing the number of broken constraints.
Overall, the lab validation yielded promising results regarding the applicability of TEASE to the setting of the iPraktikum. In the next section, we seek a deeper understanding of the results of the lab validation before releasing the tool to be used in the solution domain.

5.2 Static Validation: Focus Group Workshop

According to the Technology Transfer Model, the aim of the static validation is to present the proposed solution to the future users to gather feedback before it is deployed and piloted [GGLW06; Woh+12]. The objective of this validation step follows the objective of the lab validation of evaluating the applicability and effect of TEASE on the team initialization process. In addition to the quantitative data from the lab validation, we want to look deeper into the following research questions:

RQ3 What is the program managers' experience performing the team initialization with TEASE?

RQ4 Are there requirements that need to be fulfilled before the tool can be released?

Thus, the purpose of this validation step was both explanatory and exploratory. Since the objective was to gather qualitative data and this study did not have a confirmatory purpose, no hypotheses were formulated.
A focus group workshop was conducted just after the program managers had completed the team initialization using TEASE to be able to get a detailed account of their experience. The following sections describe how the workshop was conducted to collect the data and how the qualitative feedback was analyzed. We then summarize the results and derive implications for the further evaluation of TEASE from them. Threats to validity are discussed in Section 5.5 together with the other validation steps.

Data Collection

A focus group workshop was conducted with the program managers after they had completed the lab validation procedure described in Section 5.1. A colleague who had not been part of the team initialization moderated the workshop in the role of the experimenter, while the three program managers took part as subjects. The workshop was voice recorded and the experimenter took additional notes.
The structure of the workshop was derived from the recommended structure for agile retrospectives [DLS06] to focus on the neutral collection of data first. The workshop was conducted as follows:

Setting the Stage: The experimenter introduced the goal of the workshop and asked the participants to share as openly as possible. Questions of the program managers were clarified.

Gathering Data: The program managers were given five minutes to write down anything that comes to mind about their experience with TEASE, both positive and negative, during the team initialization to enable them to gather their thoughts. After this, the experimenter showed each step of the team initialization process again and asked the program managers to write down any additional points that they remembered. Finally, the program managers were given the comparison of the TEASE-supported assignment and the previous manual assignment and were given 15 minutes to individually reflect on where they think TEASE has helped improved the outcome, along with the reasons for the improvement. The program managers wrote the gathered points on sticky notes and put them on a whiteboard.

Generating Insights: On the whiteboard, the experimenter divided positive and negative feedback and clustered the points according to the process step or topic

they belonged to. Duplicates were removed. The program managers discussed each cluster on the board one by one, clarifying each other's opinions and generating insights on the feedback. Moreover, they decided for each note whether a requirement for TEASE could be elicited from the feedback. Requirements were written on an individual list next to the whiteboard.

Deciding what to do: As a last step, the program managers went through the list of requirements and determined their importance. Requirements were classified using priority levels *Low*, *Medium*, *High*, and *Critical*. A priority level was assigned when at least two of the three program managers were in agreement. The priority level of *Critical* was also assigned in the case of only one vote, since it was defined as a requirement without which it is impossible to perform a real team assignment with TEASE, which means that it had to be realized before dynamic validation could begin.

The voice recording of the workshop, the experimenter's notes, the state of the whiteboard, and the list of prioritized requirements were saved and used for analysis as described in the next section.

Data Analysis

Since the workshop was already aimed at eliciting the program managers' feedback and requirements, the output that they had produced in the form of the clustered feedback and prioritized list of requirements was put at the center of the analysis. The transcript and notes were used to clarify and enrich the information produced by the program managers.
The state of the whiteboard was cleaned up by removing further duplicates and overlaps between clusters. Moreover, disregarded clusters that had been deemed as unimportant during the discussion by the program managers, were too far from the research questions or were already covered in the form of elicited requirements were disregarded. The resulting clusters of feedback (excluding requirements) were the following:

Improvement: Concrete cases or process steps in which TEASE led to a positive experience as opposed to the manual process.

Duration: Explanations offered by the program managers for why the process duration of each step differed from the manual duration.

Result Interpretation: Explanations offered by the program managers regarding the difference in assignment quality using TEASE as opposed to the manual assignment.

The list of elicited requirements was ordered by descending priority. Each requirement was mapped to the corresponding process step(s) from TEASE's instantiated *Assignment Preparation* or *Team Initialization* workflows (cf. Section 4.3).
In the following, we present the results of the focus group workshop, structured by the research questions that are the basis of this evaluation step.

Results: Program Managers' Experience (RQ3)

With **RQ3**, we were interested in the experience of the program managers when performing the team initialization, as well as their qualitative explanation to the results of the lab validation.
In the *Improvement* feedback cluster, the biggest improvement the program managers identified was the elimination of most manual optimization steps. For instance, instead of counting development and test devices in each team by hand like in the manual process (cf. Section 4.2), TEASE automatically generated suggestions in which the minimum number of devices is met and also alerted them when a subsequent change broke one of the constraints so that this could not happen by accident. They also liked that while the tool visualized if an adaptation deteriorated the assignment's objective function, they could choose to perform the action regardless if they thought that it was acceptable.
When it comes to the *Duration* cluster, the program managers attributed the drastic decrease of process duration to the fact that manual optimization usually takes most time during team initialization, while an algorithm can perform this step very quickly. They hypothesized that the fact that the data was already digital could also speed up the process, but could not agree on another logical explanation for this effect.
Finally, the *Result Interpretation* cluster contained attempts to explain the difference in objective value function between manual and TEASE-supported initialization. The program managers had discussed what the reasons for the biggest improvements could be and identified two likely reasons:

(a) With 'problematic teams', i.e., teams with high values in terms of project priority in the Person Pool (such as T06 and T10), it is difficult to manually minimize the priority. In the manual process, the program managers staff these teams first to ensure they find the best possible solution. However, TEASE still found a more optimal assignment for these teams.

(b) In some teams with a favorable overall priority (such as T08), TEASE still provided an improvement of 0.5 or larger. The program managers inspected the two assignments and found that while they were almost the same, they had assigned one student who had given this project their second choice to another

project during the manual initialization. Thus, in this case TEASE eliminated a manual mistake.

Results: Requirements (RQ4)

With regard to **RQ4**, the requirements that emerged are shown in Table 5.6 including the mapping to the TEASE process step(s) they correspond to. A priority level of *Critical* was defined to mean that at least one program manager doubtful that the team initialization process could be completed only in TEASE without it, thus the requirement had to be realized before dynamic validation could begin.

Table 5.6: Requirements elicited by the program managers during static validation

ID	Requirement	Priority	Activity
R1	The user should see the following additional information on each card without needing to open it: first four project priorities, development and test devices, language level.	Critical	Review/Manually Adapt Course Assignment
R2	The user should be able to add constraints on a per-team basis. A team-based constraint should override the global constraints.	Critical	Define/Refine Assignment Problem
R3	In addition to the overview over broken and satisfied constraints, the user should see a graphical overview of the project priority distribution.	High	Review/Manually Adapt Course Assignment
R4	The user should see Instructor Ratings in increasing order on each student's card: *Novice – Intermediate – Advanced – Expert.*	Medium	Assign Instructor Rating to all Students

Discussion & Implications

With regard to the research question **RQ3**, we can conclude that the program managers perceived the biggest improvements when it comes to the reduction of manual work through TEASE. They attributed the reduced duration of the process as well as the increase in assignment quality largely to the capability of TEASE to take many, even conflicting, factors into account and find an optimal solution much more quickly and accurately than the manual process.
Looking at the data from the run of TEASE in the previous two semesters, we found further examples for the two possible reasons for improvements the program managers identified: a) applies to T19, T29, and T30, while b) likely occurred with T13, T14, T27, and T28. These effects are also visible in Figure 5.4. The program managers discussed that while it is theoretically possible to reach the same result with the

manual process, TEASE was not only more likely to find it, but did so faster than humanly possible.
The investigation of **RQ4** revealed two critical requirements that need to be realized before the tool can be used in the dynamic validation. All identified requirements were implemented and released as part of Version 1.0 of TEASE, which was used in the next evaluation step. In summary, the static validation provided some further qualitative explanations to the quantitative data from the lab validation.

5.3 Dynamic Validation I: Action Research and Iterative Improvement of TEASE

After having evaluated the applicability of TEASE in the setting of the iPraktikum and having examined its effect on the existing process, the goal of the dynamic validation was to improve the tool's usefulness from the perspective of the program managers of the multi-project course. To pursue this goal, the following research questions were investigated:

RQ5 Can TEASE be tailored to cover all activities and requirements of the existing team initialization process?

RQ6 Which new requirements emerge beyond the original process?

While **RQ5** served to iteratively develop the tool to improve its tailoring to the existing process, **RQ6** was aimed at eliciting requirements beyond this process to make the team initialization more manageable than before. For **RQ5**, the following hypothesis were derived:

H5.1 The amount of requirements that are qualified as gaps to the original process decrease with each iteration of TEASE (**RQ5**).

The research questions and hypothesis were investigated in a three-part case study in the iPraktikum: The team initialization was performed using TEASE for three subsequent instances of the course and followed up with a focus group workshop with the program managers after each iteration. The elicited requirements were classified as either process gaps or improvements beyond the existing process. We used action research, which means that the aim was to actively improve the tool using the elicited, prioritized requirements that were implemented and released as a new TEASE version in-between iterations.
In the following, we describe the data collection by summarizing the team initialization process of each course instance and explain how the data from the focus group

workshops was analyzed. After presenting the results, we discuss them with regard to the research questions. Threats to validity are addressed in Section 5.5 together with the other validation steps.

Data Collection

Following the lab and static validation, the feedback and requirements elicited from the program managers were implemented and released as TEASE V1.0. In the subsequent three instances of the course, the program managers of the iPraktikum performed the whole process using TEASE as described in Section 4.3.
After each run, feedback was collected from the program managers in the same focus group workshop format as described in Section 5.2. The resulting data was a prioritized list of elicited requirements as well as a clustered body of textual feedback from the program managers, along with recordings of the workshop that could be used to clarify the results where necessary.

Course Data & Assignment Problem Configurations

In the following, we describe the team initialization process in terms of Person Pool statistics, resulting constraints and assignment quality measured by objective function value and number of broken constraints. An overview over the course size and results is shown in Table 5.7.

Table 5.7: Assignment quality in three iterations of team initialization with TEASE

	Winter 2017/18	**Summer 2018**	**Winter 2018/19**
Person Pool size	78	74	85
Project Pool size	12	9	10
Avg. priority	2.08	1.85	1.62
Objective function F_T	25.01	16.69	16.19
Broken constraints $\lvert\overline{C}_T\rvert$	2	1	1

The three instances of the course had a size of 74-85 students and 9-12 project teams. The resulting assignment had a mean objective function value between 1.62 and 2.08, averaged over all teams in the course. The summated assignment objective function value F_T (cf. Section 5.1) was between 16.19 and 25.01. These metrics are explained in Section 5.1 and are comparable to the results of the first lab validation using historical data.
While TEASE found a solution without breaking any constraints, the program managers accepted a few broken constraints to satisfy team-specific requirements such as a client asking to work with a particular student. In total, the program managers broke

two *female student* constraints in the winter 2017/18 instance of the course, one *development devices* constraint in the summer 2018 instance, and one *Novice* constraint in the winter 2018/19 run.
The constraints were again derived from the statistics of the Person Pool, which are shown in Table 5.8 for all three iterations, including the total numbers present in the Person Pool for each criterion, the corresponding average over the number of teams, as well as the lower and upper bound for the selected constraint. These numbers do not give an indication of the quality of the resulting assignment, but are included here to describe the Person Pool and the approach of the program managers in each iteration.

Table 5.8: Person Pool statistics and global constraints in three iterations of team initialization with TEASE (values marked with * have team-based exceptions)

	Winter 2017/18				**Summer 2018**				**Winter 2018/19**			
	Total	**Avg.**	**Min**	**Max**	**Total**	**Avg.**	**Min**	**Max**	**Total**	**Avg.**	**Min**	**Max**
Students	78	6.5	6	7	74	8.2	7*	8*	85	8.5	8	9
Expert	12	1	1*	2	5	0.56	0*	1	11	1.1	1	2
Advanced	17	1.42	1*	2*	15	1.67	1*	2*	21	2.1	2	3
Intermed.	30	2.5	-	-	32	3.56	-	-	37	3.7	-	-
Novice	19	1.58	1	2	22	2.44	2*	3*	16	1.6	1	2
D. Dev.	61	5.08	4*	-	42	4.67	4*	-	48	4.8	4	-
T. Dev.	55	4.58	4	-	43	4.78	4*	-	57	5.7	4	-
Females	13	1.08	1	-	15	1.67	1	-	17	1.7	1	-

In the first two runs, both global and team-based constraints were set, while the third run was possible to perform with only global constraints. In the winter 2017/18 instance, the program managers relaxed the *Expert* constraint for two teams that were prioritized lower than the others to be between 0 and 2. In order to ensure a skill balance in these two teams, the *Advanced* constraint was raised to be between 2 and 3. In the following summer 2018 instance, the program managers were required to initialize a bigger team. They set the team size to be between 12 and 13 and adjusted the remaining constraints to account for the larger team size. The resulting constraints for this team were: *Expert* at least 1, *Advanced* between 2 and 3, *Novice* between 3 and 4, and development and test devices with a minimum of 6 each.

Data Analysis

The data was analyzed similarly to the process described in Section 5.2, resulting in a prioritized list of elicited requirements mapped to the corresponding process step from the TEASE workflows. In addition, each requirement was classified into one of these two types:

Process Gap (GAP): A requirement that constitutes an activity or feature of the original process that is not being met by the current version of TEASE.

Requirement beyond original process (NEW): A requirement that goes beyond the original process, but is seen as useful by the program managers.

The resulting requirements were used to improve TEASE and release a new version. If there were too many requirements to implement during a single semester, the ones that were prioritized most highly by the program managers were selected. The following sections summarize the requirements elicited from each run of the evaluation, including when they were realized and integrated into TEASE. Instead of structuring the results according to the research questions like in the previous sections, they are structured chronologically by iteration.

Results: Iteration I (Winter 2017/18)

The TEASE-supported team initialization process in the winter 2017/18 instance of the iPraktikum resulted in 11 new requirements as shown in Table 5.9. Six of these new requirements were deemed to be critical by at least one program manager.
The *Type* column shows whether the requirement is classified as a current process gap (*GAP*) or a requirement that goes beyond the original process (*NEW*). When there is no classification, the requirement refers to an issue that was caused by a feature of TEASE itself, so that it cannot be connected to the original process. In total four out of the 11 elicited requirements were classified as process gaps, five were new requirements, and two could not be classified.
The new requirements discovered in the first run can be clustered as follows:

User interface and user experience: these requirements concern showing existing data differently. For instance, the first iteration revealed that the current UI limits the ability of program managers to iterate over constraints and Instructor Ratings. R5, R6, R10, R11, and R13 fall into this category.

Additional data: this group contains the need for new data to be visualized or taken into account when generating suggestions, such as R7, R8, or R9.

New Features: these concern new functionality to improve the convenience of the tool, namely R12, R14, R15.

Following the first run, four out of the six critical requirements as well as two non-critical ones were implemented and released as TEASE V2.0.

Table 5.9: Requirements discovered in the winter 2017/18 instance of the iPraktikum (TEASE V1.0). Requirement IDs are continued from Table 5.6.

ID	Requirement/Issue	Priority	Type	Released
R5	The user interface is built so that the process has to be sequential, and it is difficult to jump between steps.	Critical	—	V2.0
R6	More details needed on cards: all priorities, iOS experience, Instructor Rating as label.	Critical	GAP	V2.0
R7	Instructor Ratings and skills should be on the same scale.	Critical	GAP	V2.0
R8	A new type of device (supporting ARKit) is needed both in the UI and in the algorithm.	Critical	NEW	V2.0
R9	Tutors' comments regarding the student's performance in the Bootcamp should be shown for each person.	Critical	GAP	V2.1
R10	An overview is needed over statistics of the Person Pool (totals and averages for all criteria; priority distribution of projects in course) to be able to set constraints faster.	Critical	NEW	V2.2
R11	Instructor Ratings and skills should use the same colors.	High	GAP	V2.0
R12	It should be possible to automatically order all team members by descending Instructor Rating to better see the skill level.	High	NEW	V2.1
R13	The Instructor Rating as dotted border is difficult to see for color-blind program manager.	Medium	—	V2.0
R14	It should be possible to navigate between persons using the arrow keys of the keyboard.	Medium	NEW	V2.1
R15	It should be possible to rate persons using the number keys of the keyboard.	Medium	NEW	V2.1

Results: Iteration II (Summer 2018)

TEASE V2.0 was used to conduct the team initialization in the summer 2018 instance of the iPraktikum. The resulting new requirements are shown in Table 5.10 including their categorization into a *NEW* or *GAP* type according to the research questions. The program managers identified one process gap and three new requirements.

Two out of the three critical requirements concern additional data needed to improve the picture of each student that the program managers get when assigning Instructor Ratings: In order to better be able to assess the students, the program managers want to see their self-reported interests next to their skills, as well as their assessment of the Swift Bootcamp (cf. Section 4.1) next to the tutor's assessment. Moreover, the

Table 5.10: Requirements discovered in the summer 2018 instance of the iPraktikum (TEASE V2.0). Requirement IDs are continued from Table 5.9.

ID	Requirement/Issue	Priority	Type	Released
R16	Interests should be added to the detailed view, shown directly below skills using the same color scheme.	Critical	GAP	V2.1
R17	Students' own assessment of the Bootcamp should be shown alongside the tutor's assessment.	Critical	NEW	V2.1
R18	The graphical team overview as well as screenshots of the details of each person on the team should be exported automatically to forward to the project leaders.	Critical	NEW	V2.2
R19	The following actions of program managers should be logged: change of constraints, re-distribution, manual change.	High	NEW	V2.2

program managers discovered that automating two additional process steps, namely the documentation of changes as well as the exporting of the finished graphical team overview, would shorten the process even more.
After this second run, implemented some of the requirements that were left over from the winter 2017/18 iteration were implemented along with R16 and R17 for the subsequent third iteration with TEASE V2.1.

Results: Iteration III (Winter 2018/19)

V2.1 of TEASE was used for the last dynamic validation, the results of which are shown in Table 5.11. This iteration resulted in only two new requirements, one of which was deemed critical by at least one program manager. R21 was classified as a process gap because assigning a particular student to a team is possible in the manual process.
Both R20 and R21 concern manual changes to the assignment: the program managers want an easier way to localize students with specific attributes, and they want to be able to manually pre-assign a student to a team.

Table 5.11: Requirements discovered in the winter 2018/19 instance of the iPraktikum (TEASE V2.1). Requirement IDs are continued from Table 5.10.

ID	Requirement/Issue	Priority	Type	Released
R20	The user should be able to filter the Person Pool or the initialized teams according to: interest level in category, skill level in category, Instructor Rating.	Critical	NEW	V2.2
R21	It should be possible to 'pin' a person to a team before distribution, ensuring that this person is assigned to that team.	High	GAP	V2.2

Results: Evolution of TEASE

By using action research, the focus was on iteratively improving TEASE according to the feedback of the program managers. This involved major changes to the user interface of the tool.

To illustrate some of these changes, Figure 5.5 shows the evolution of the student cards, while Figure 5.6 illustrates improvements in the way student data is displayed and Figure 5.7 shows the changes in the team statistics UI.

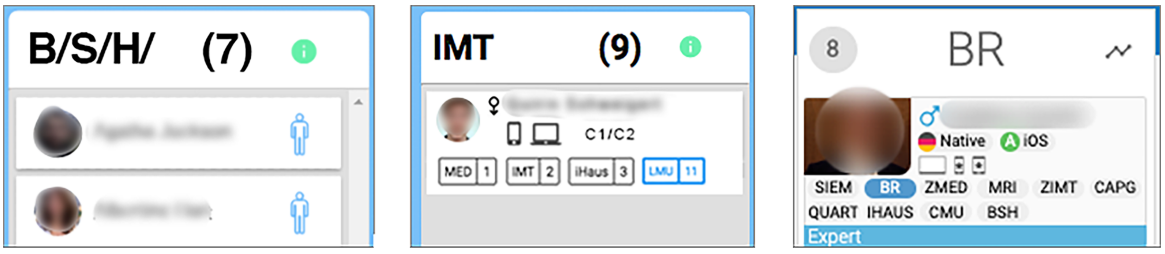


Figure 5.5: Evolution of TEASE UI: Student cards

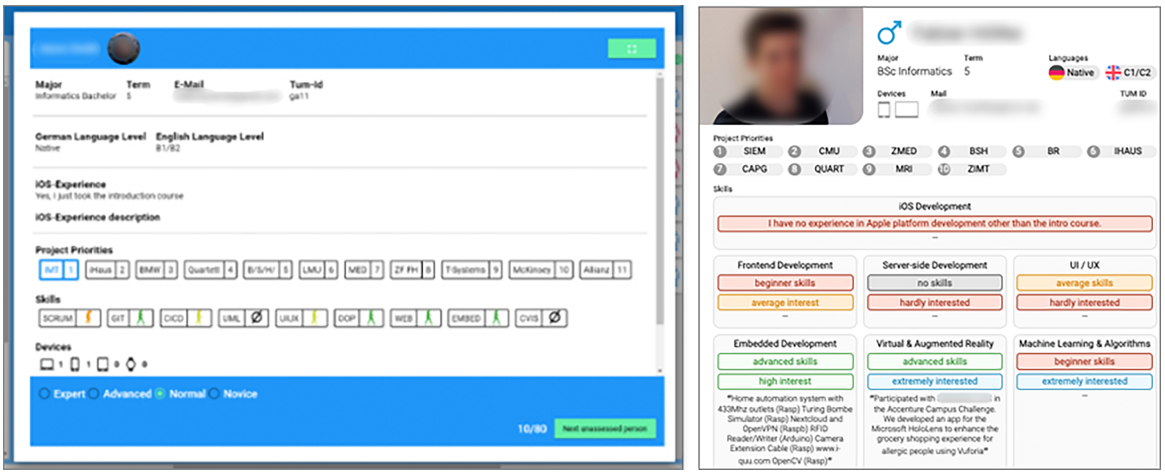


Figure 5.6: Evolution of TEASE UI: Student details

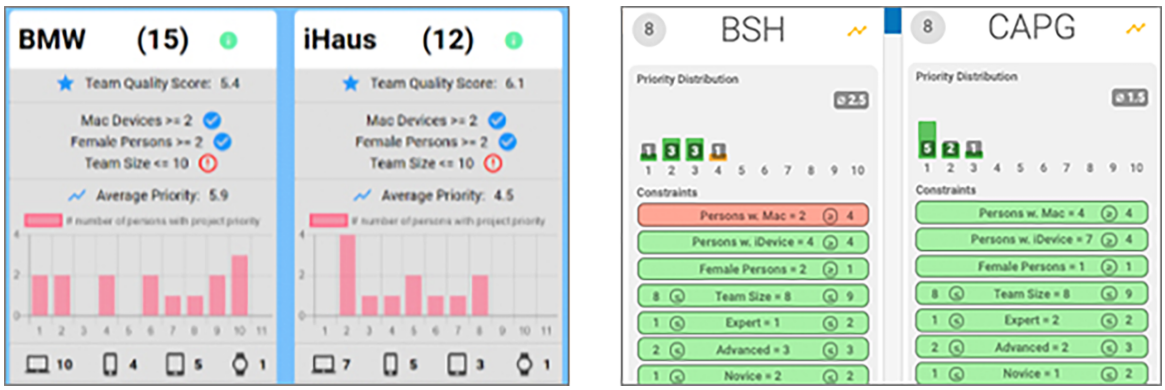


Figure 5.7: Evolution of TEASE UI: Team statistics

Results: Requirements Evolution over all Iterations (RQ5 and RQ6)

Table 5.12 shows the evolution of newly discovered requirements including their categorization into process gaps (*GAP*) or new requirements (*NEW*) over the three iterations of improvement of TEASE. Moreover, the table includes the number of existing unrealized requirements that have been elicited in a previous focus group workshop, but not realized until the next iteration.

Table 5.12: Summary of new and existing unfulfilled requirements

	TEASE version	**New requirements**			**Existing unrealized requirements**		
		Total	**GAP**	**NEW**	**Total**	**GAP**	**NEW**
Winter 2017/18	V1.0	11	4	5	0	0	0
Summer 2018	V2.0	4	1	3	5	1	4
Winter 2018/19	V2.1	2	1	1	3	0	3
	V2.2	–	–	–	0	0	0

The number of newly discovered requirements went from 11 to four to two. Moreover, we were able to reduce the number of existing process in-between iterations as well as the number of new process gaps discovered, which went from four in the first run to one in each subsequent iteration. Thus, we obtain support for **H5.1**. In the following section, these results are discussed with regard to the research questions.

Discussion & Implications

The program managers were able to perform the team initialization for three subsequent instances of the iPraktikum. With regard to **RQ5**, we obtained support for **H5.1** and further support from the qualitative feedback of the program managers that each iteration of TEASE turned out to be a better fit to the original process. Moreover, both the quantity and priority of newly discovered requirements declined over the three iterations. While this shows that TEASE could be tailored to the process in the iPraktikum, more research is needed to examine whether it can be just as well tailored to other settings.

In addition to tailoring the tool to the existing process, **RQ6** looked beyond it to see what other requirements emerged that can make the team initialization even more manageable for the program managers. Nine requirements that were classified as new were elicited and integrated into TEASE. Most of these requirements concerned additional data that should be integrated into the student cards or team and person pool statistics, or functionality to quickly compute metrics or locate students which is not possible in the manual process. Thus, we were able to extend the existing process through TEASE and make the team initialization more convenient for the program managers.

While improving the experience of other stakeholders than the program managers was not part of the research questions for this validation step, it is also worth mentioning that the exported team visualizations were forwarded to the project leaders after each initialization so that they could get a first overview of their team. We received positive feedback and anecdotal evidence about the usefulness of the visualization of students' skills, interests and experience to instructors who are not directly responsible for the team assignment. The following second dynamic validation explores the perception of TEASE from the perspective of a project leader in more detail.

5.4 Dynamic Validation II: Observational Study with Domain Experts

In this last validation step, an observational study was conducted to evaluate the usability of and attitude toward TEASE from the point of view of domain experts who have never used the tool before. To reach this goal, the following research questions were formulated:

RQ7 Can new users successfully perform the team initialization with minimal prior training?

RQ8 What issues do the participants have when using the tool and how challenging do they find the tasks?

RQ9 Are the participants satisfied with TEASE as a solution for performing the team initialization?

According to Nielsen's classification of usability, we evaluated the attributes *Learnability*, *Satisfaction*, and *Errors* [Nie94].
Learnability refers to the ability of the user to rapidly start performing some work with a system, which Nielsen qualifies as the most fundamental usability attribute. Grossmann et al. [GFA09] derive a taxonomy for learnability from multiple definitions, with the dimensions *learnability scope* (single or multiple usage periods) and *user definition* (level of experience with computers in general, with the interface itself, or with similar software; level of domain knowledge). They also make recommendations regarding performance measures. **RQ7** investigates the initial learnability of TEASE, i.e., the ability of a user to perform well during an initial set of tasks. In this case, we examined users whose experience with the tool was non-existent, but who had domain knowledge in team factors in student teams, having led and coached multiple teams before the study. Metrics for performance and corresponding hypotheses are derived in the next section.
According to Nielsen, error rate of the system should be low, and errors the user does make should be easy to recover from. It is especially important to avoid catastrophic errors that either go unnoticed and end up in the final product, or that destroy the user's work. With **RQ8**, we examined the users' errors, but also went beyond this criterion and looked into how much help was needed to complete a set of tasks. Moreover, we explored the users' perceived level of challenge of the tasks.
Finally, **RQ9** investigated satisfaction, a usability attribute describing how pleasant it is to use the system. Since it is a subjective measure, Nielsen recommends to elicit it by directly asking users for their opinion, e.g. by filling in a questionnaire after using

the system and averaging their responses into "an objective measure of the system's pleasantness" [Nie94]. We explored the users' attitude toward TEASE, amongst others by investigating their perceived usefulness and perceived ease of use as well as their free-text comments about using the tool.

We did not examine Nielsen's other two attributes, *Efficiency* and *Memorability*. *Efficiency* measures the productivity of users when they have learned to work with the system, which had already been covered in previous evaluation steps, amongst others in comparison to the manual team initialization. *Memorability* refers to the performance of the user when returning to the system after not having used it for a period of time, which was not the goal of this validation step.

The following sections describe the study design, hypotheses, subjects, instructions, and survey instrument. We then report the data collection and analysis activities, followed by the results structured by the research questions. Finally, the results are interpreted and implications for future research are derived. Threats to validity are discussed in Section 5.5 together with the other validation steps.

Study Design

Figure 5.8 shows an overview of the study design for this validation step. In order to investigate the above-described research questions, we conducted an observational study using the think-aloud protocol, which entailed asking the subjects to solve a problem while saying their thoughts out loud as they come to mind [RH09b; VBS94]. We chose this approach because it has shown to yield direct, rich qualitative data about the thoughts in the users' minds while not disturbing the problem-solving process, and it has been proven to work well with users with advanced domain knowledge [NCY02; VBS94]. Moreover, the approach can be combined with a questionnaire, which were used to answer the remaining research questions (see Appendix D.4 for the full questionnaire).

In order to investigate **RQ7**, a definition of 'successful performance' in the context of the study was needed. From Grossman et al.'s suggestions of learnability metrics, we chose two metrics based on task performance, namely *percentage of users who complete a task optimally* and *percentage of users who complete a task without any help* [GFA09]. In terms of 'optimal completion' of a task, we measured how subjects configure the assignment problem by setting minimum and maximum values for its constraints when they performed the team initialization, since this activity is the biggest determinant of the resulting assignment. While there is no single optimal solution to configure constraints, we found that there is a range within which the constraints should be set for the suggestions to be of best quality considering the Person Pool at hand. Thus,

the following hypotheses were derived for this research question, both of which were intended to be answered using data from the think-aloud study:

H7.1 At least 50% of users define all constraints $c_i \in C$ with bounds $c_{i_{min}}, c_{i_{max}}$ in the acceptable range $[l_i, u_i]$.

H7.2 At least 50% of users complete six or more tasks without asking for assistance, excluding questions on the task instructions themselves.

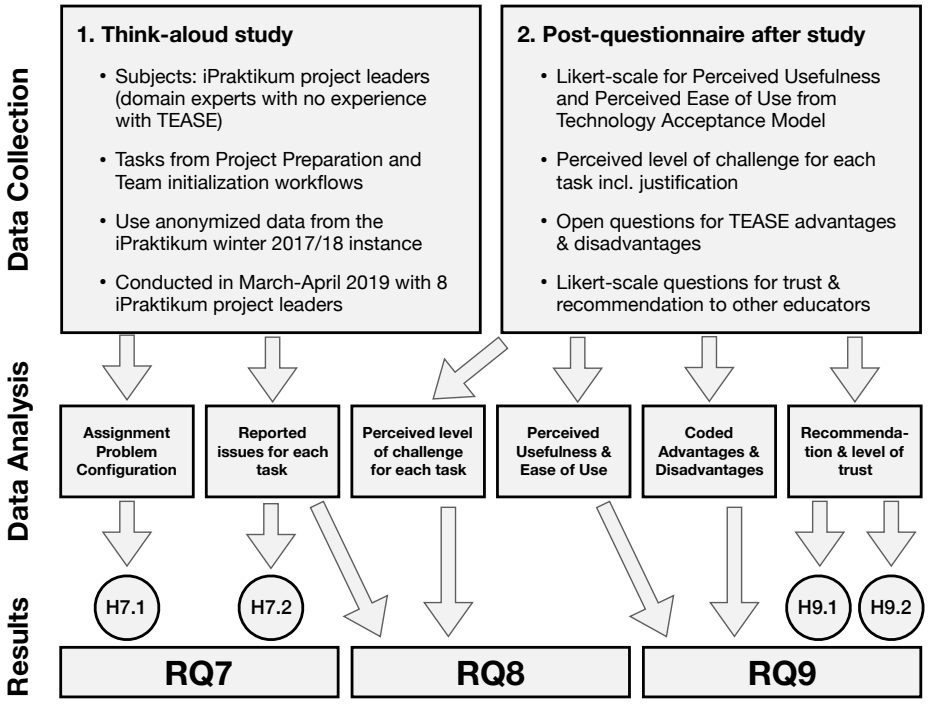


Figure 5.8: Study design of dynamic validation II

The issue-centered aspect of **RQ8** was addressed using the categorized reported issues from the think-aloud study, while the users' perceived challenge for each task was elicited from the responses of the questionnaire.
The questionnaire also investigated **RQ9** through Likert scales measuring perceived usefulness and perceived ease of use, two influencing factors for technology usage and acceptance. Perceived usefulness is the extent to which a user believes a system will help them perform their job better and perceived ease of use is "the degree to which a person believes that using a particular system would be free of effort" [Dav89]. While no hypotheses were formulated for these two measures, we complemented them with users' answers to open-ended questions regarding TEASE's advantages and disadvantages and used the results as a first indicator of users' satisfaction with the tool. In addition, the following hypotheses were formulated for **RQ9**:

H9.1 The majority of users claims they trust TEASE to support them in the team initialization ('Agree' or 'Strongly Agree').

H9.2 The majority of users would recommend TEASE to fellow educators managing multi-project courses ('Agree' or 'Strongly Agree').

Subject Selection

The target group for this study were the project leaders of the iPraktikum. The project leaders are doctoral candidates or lecturers who are responsible for management and teaching activities in the project (cf. Section 4.1).
Three criteria were defined that should hold true for study subjects:

(a) Has led a project in the iPraktikum in the past semester,

(b) Has led a minimum of three iPraktikum projects in the past, and

(c) Has never performed the team initialization, neither manually nor using TEASE.

At the time of recruitment for this study in March 2019, 11 project leaders at the Chair for Applied Software Engineering fulfilled all criteria. Having led multiple projects in the iPraktikum, they had experienced which factors are important to initialize a well-functioning project team in this context. Therefore, they could be considered domain experts in team composition for project courses, with no experience with performing the process itself.
All candidates were invited to take part in the study in March 2019. Out of the 11 contacted candidates, eight agreed to participate, constituting the participant sample for this study.

Task Selection

For the selection of tasks for a think-aloud study, Van Someren et al. [VBS94] recommend to choose tasks depending on the subjects' skill level: They should be challenging enough so that the users can complete them almost without thinking, but not so challenging that they cannot accomplish them on their own. Moreover, the tasks should be representative of the work with the system, i.e., not portray an artificial or unusual problem, and one should pick a small set of problems to solve to avoid the user getting tired or fed up with the study [VBS94].
A subset of tasks was selected from the original TEASE-supported *Project Preparation* and *Team Initialization* workflows described in Section 4.3 and ran a pre-test of the study with one of the iPraktikum program managers to determine if the instructions are understandable and the quantity and difficulty of tasks were suitable for the

study. The resulting tasks were similar to the ones performed during static validation (cf. Section 5.2) and are shown as graphical process in Figure 5.9 and in a textual form in Table 5.13. The task descriptions handed out to the participants are available in Appendix D.3.

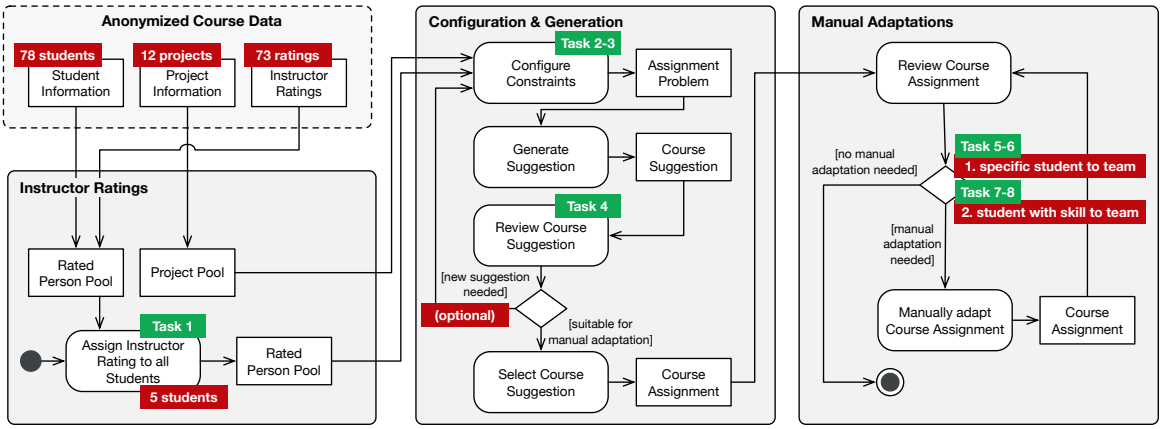


Figure 5.9: Task progression during think-aloud study

To obtain realistic data, we used anonymized student and project information from the iPraktikum winter 2017/18 instance with 78 students and 12 projects. The original *Instructor Ratings* assigned by the program managers were imported into TEASE, letting the participant start with a *Rated Person Pool*, but excluded the ratings for five students from the data.

The first task was to *assign Instructor Ratings* to these students using the self-reported information available in TEASE and comparing the data to the other, already rated, students to create consistency (Task 1 in Figure 5.9). After this step, the participants were directed to the statistics of the `Rated Person Pool` (cf. Figure 4.10) to `configure constraints` for the suggestion algorithm. In order to make this step easier to understand, the instructions were split into constraints for team size, devices, and female students (Task 2) and constraints for team balance (Task 3). The latter task required the participants to select three out of the four Instructor Rating levels to set constraints for, letting the algorithm distribute the remaining category.

After setting the constraints, the participants were instructed to `generate a suggestion` and `review the suggestion` concerning the given criteria (Task 4). Moreover, they were free to refine the assignment problem and `re-generate a suggestion` as often as they wanted, although they were not actively instructed to consider doing so. After `selecting a suggestion`, there were two `manual adaptations` to perform, each split into two tasks of first performing the change and then evaluating its effects on the assignment. The first manual adaptation consisted of moving one specific student of the course into a given team, provided they were not already assigned to

that team (Task 5). If a change was performed, the participants should `review the course assignment` and evaluate the effect of the change on the given constraints, and if constraints were broken, attempt to manually fix at least one (Task 6). The last adaptation concerned the addition of any student with advanced Machine Learning skills and a minimum Instructor Rating of *Intermediate* to a given team (Task 7). Participants were instructed to use the highlighting feature of TEASE to locate possible candidates and move a student whose project priority is not decreased by more than three into the team. Finally, they were asked to reflect on the impact of the change on the quality of the assignment, measured by the previously set criteria; if they identified weaknesses, they were asked argue how they would attempt to fix them without performing further manual changes (Task 8).

Table 5.13: List of tasks for observational study (see Figure 5.9 for graphical process)

Task #	**Task Content**
Task 1	Assign Instructor Rating to the five unrated students
Task 2	Configure constraints for team size, development and test devices, female students
Task 3	Configure constraints for Instructor Rating balance (3/4 Instructor Rating levels)
Task 4	Generate a suggestion and review it regarding the previously configured criteria
Task 5	Move given student to specific team (if applicable)
Task 6	Evaluate impact of change on criteria and fix broken constraint (if applicable)
Task 7	Locate student with interest in given area and move to specific team
Task 8	Evaluate impact of change on criteria (if applicable)

Questionnaire Development

The questionnaire was designed to obtain responses corresponding to the data shown in Figure 5.8. The full questionnaire is available in Appendix D.4, and the mapping of questions to codes as well as the aspect measured by each question are shown in Table 5.14.
The first question lists all the tasks and asks participants how challenging they found each of them. Responses are recorded on a 5-point ikert-style scale from 'Not challenging at all' to 'Extremely challenging'. For each task that was marked 'Challenging' or 'Extremely challenging', participants are asked to explain their responses in question 2. Following a 5-point standard Likert-style question of whether participants would recommend TEASE to fellow educators of multi-project courses, they were asked to list advantages and disadvantages of using the tool in a free-text form.
Finally, perceived usefulness and perceived ease of use were measured based on the scales developed by Davis, who has suggested and evaluated a 6-item Likert scale for each [Dav89]. One item was omitted from each scale, resulting in two 5-item Likert scales in the questionnaire as shown in Appendix D.4.

Table 5.14: List of questions in the observational study questionnaire

#	Code	Type	Aspect
1	PC1	Likert	Perceived challenge
2	PC2	Open-ended	Perceived challenge
3	RETR	Likert	Recommendation & Trust
4	ADV	Open-ended	Advantages
5	DADV	Open-ended	Disadvantages
6	PU	Likert	Perceived usefulness
7	PEOU	Likert	Perceived ease of use
8	COMM	Open-ended	Additional comments (optional)

Data Collection

The think-aloud study was conducted with eight participants between March and April 2019. Before each run, TEASE in its newest Version 2.2 was prepared using the above-described Person and Project Pool on a machine designated for this study. We followed the recommendations of Van Someren et al. concerning the setting and preparation [VBS94]. Participant were informed about the voluntary nature of the study and signed a consent form before they began (cf. Appendix D.1).
Each participant was first given a one-pager explaining the context of the study, their now-assumed role of program manager and the criteria on which they should base the team composition (cf. Appendix D.2). Following this, the experimenter gave them a short overview over TEASE and its user interface. The tool was presented without connection to the tasks or advice on how to complete them, explaining that they would receive task instructions after the introduction. The experimenter only answered general questions regarding the user interface, inquiries on how to best perform the team initialization were not answered to ensure better comparability of the results. Moreover, the experimenter explained the think-aloud protocol and instructed each participant to say everything that goes through their head out loud. The participants were told to try to solve problems on their own first, but if they could not proceed, say the name of the experimenter and then ask their question; this was necessary to ensure that the experimenter did not address a question that was asked out loud as part of the think-aloud process.
Following the introduction, the experimenter started the screen recording and additional voice recording and the participants received the task sheet available in Appendix D.3. While the participants worked through the tasks, the experimenter took notes of challenges and issues that occurred. When the participants fell silent or forgot to share an aspect of their approach, the experimenter prompted them to keep talking out loud, but did not interfere unless a participant asked for advice or they had misunderstood a task so far that it no longer made sense to perform it. Each intervention was noted down in the study log.
After having gone through the tasks, the participants were given the questionnaire (cf. Appendix D.4) to fill out immediately. Finally, the experimenter debriefed each participants by thanking them for their contribution and answering any additional questions that had come up. The final assignment in TEASE was exported and saved along with the questionnaire data and study recordings.

Data Analysis

As shown in Figure 5.8, the data that was derived from the execution of the tasks in TEASE consists of the exported assignment problem configuration and the recorded issues of participants. The exported assignment problem configurations, made up of values set for seven of the eight possible constraints, were compared to what the program managers consider to be an acceptable range for constraints with the current course, as shown in Table 5.15. The values were derived from the strictest possible settings considering the values of the criteria in the current Person Pool. For instance, with 61 development devices present in the course, the strictest possible value for the constraint with 12 teams is five devices per team, but a setting of four for this constraint would also lead to an acceptable result. The compiled configurations are shown in Appendix D.5.

Table 5.15: Optimal and acceptable settings for constraints in observational study

Constraint c_i	**Optimal values**		**Acceptable values**	
	$l_{i_{opt}}$	$u_{i_{opt}}$	l_i	u_i
Team Size	6	7	6	8
Instructor Rating: Expert	0	1	0	1
Instructor Rating: Advanced	1	2	1	3
Instructor Rating: Intermediate	2	3	1	4
Instructor Rating: Novice	1	2	1	3
Development Devices	5	–	4	–
Test Devices	4	–	3	–
Female Students	1	–	1	–

The issues reported by participants were cleaned by removing questions aimed at clarifying task instructions and then classified using the following four out of the five categories of learnability issues suggested by [GFA09]:

Understanding Task Flow: The user knows that a series of things needs to be done to complete the task, but they do not know where to start or how to go about it.

Awareness of functionality: The user is not aware that a specific feature or operation exists.

Locating functionality: The user is aware of a specific feature or operation, but does not know where to find it in the user interface.

Understanding functionality: The user is aware of a feature or operation and can find it, but does not know how to use it.

The survey data was analyzed depending on the type of question: The Likert-style questions were plotted to observe and interpret the distribution of responses. For the open-ended questions regarding advantages and disadvantages of TEASE, we applied provisional coding [MHS13] using an initial set of codes derived from the advantages mentioned during static validation (cf. Section 5.2).

Results: New Users' Performance Initializing Teams (RQ7)

Participants' approach to the tasks varied notably in two places in the process, namely in the configuration of constraints and in the sequence of defining the assignment problem and generating a suggestion. When it comes to configuring constraints, the participants P01, P03, P04 and P05 used a strict approach close to the optimal values shown in Table 5.15, while the others were more liberal with the settings (cf. Appendix D.5). Moreover, half of the participants configured all constraints of the assignment problem one after the other and generated a solution either exactly once, or once for each of the two configuration tasks (Task 2 & 3). P02, P04, P05 and P07 took an iterative approach where they first set a few constraints, then generated a suggestion and went back to refining the assignment problem based on the results they obtained. This led to the re-generation of a suggestion up to nine times before transitioning to Task 5. The participants who took this approach argued that they wanted to see the effect of their settings and figure out how the constraints influence the solution, e.g. to know which of the three levels of Instructor Ratings they should set constraints for to obtain a favorable skill balance in the suggestion.
In terms of the two hypotheses, the two metrics *percentage of users who complete a task optimally* in terms of setting constraints in the pre-defined acceptable range, and *percentage of users who complete a task without any help* were selected to measure the quality of users' output. Table 5.16 summarizes the results regarding these two metrics.

Table 5.16: Assignment problem configuration quality and number of tasks completed without assistance for all participants of the observational study

	P01	**P02**	**P03**	**P04**	**P05**	**P06**	**P07**	**P08**
Constraints within $[l_i, u_i]$	7	3	7	7	6	3	5	5
Tasks without assistance	6	5	8	6	7	7	6	5

A constraint was within the acceptable range if both its lower and upper bound were within $[l_i, u_i]$ as defined in Table 5.15. Only three participants set all of the seven

constraints within the acceptable range, which amounts to 37.5% of participants. Half of the participants set a low minimum for test and/or development devices, and P06 and P08 set the minimum team size to only five, which resulted in several teams that were smaller than the average. The rest of the constraints concerned skill balance. Thus, we cannot support **H7.1**.

It is worth noting the participants who took an iterative approach to generation as described above did not all end up with a better-quality configuration despite evaluating the result multiple times. Moreover, we noticed when reviewing the study notes that participants were often able to identify the weaknesses of their assignment and argue how it could be improved, but unsure how to obtain a better result by adapting constraints.

Regarding the proportion of tasks without assistance, 75% (all but two) participants completed at least six out of the eight tasks without needing help from the experimenter. Thus, the evaluation lends support to **H7.2**.

Results: Issues and Level of Challenge (RQ8)

After removing the issues centered around clarifying task instructions, a total of 20 issues were reported by participants: One issue with Task 1, two issues for Task 2, five issues with Task 3, Task 4 with the highest number of eight issues, and Tasks 6-8 with four issues in total. Participants needed most help with the `Review Course Suggestion` step, with questions centered around how they should interpret the results or whether they were allowed to re-generate suggestions based on improved constraints.

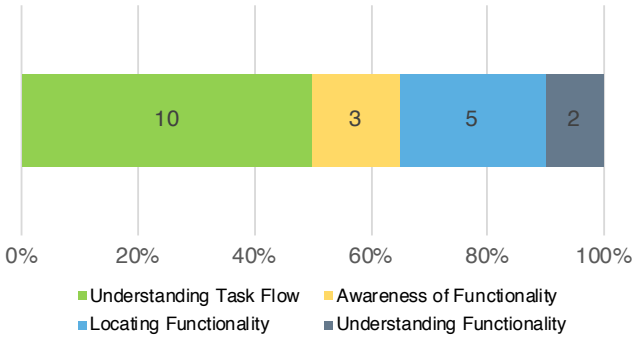


Figure 5.10: Types of issues reported by participants (20 issues reported by 8 participants)

In terms of types of issues, they were labeled according to the above-described four categories. The results are visualized in Figure 5.10. There were some problems (25%) locating functionality such as Person Pool statistics and team assignment statistics as well as the sorting feature. Half of the issues revolved around understanding task flow, with an overwhelming amount of questions about how to refine the assignment. The

remaining 25% of issues concerned awareness and/or understanding of functionality, again almost exclusively around team or Person Pool statistics.
With regard to the perceived level of challenge, the questionnaire results for the *PC1* question are shown in Figure 5.11, ordered by stated level of challenge. Task 4 (generating and reviewing a suggestion) was perceived as most challenging, which is in line with the highest reported number of issues during the study. In the open-ended question *PC2*, participants explained that it was difficult to understand the large amount of information and how the criteria influence each other, as well as spot possible trade-offs between the criteria.

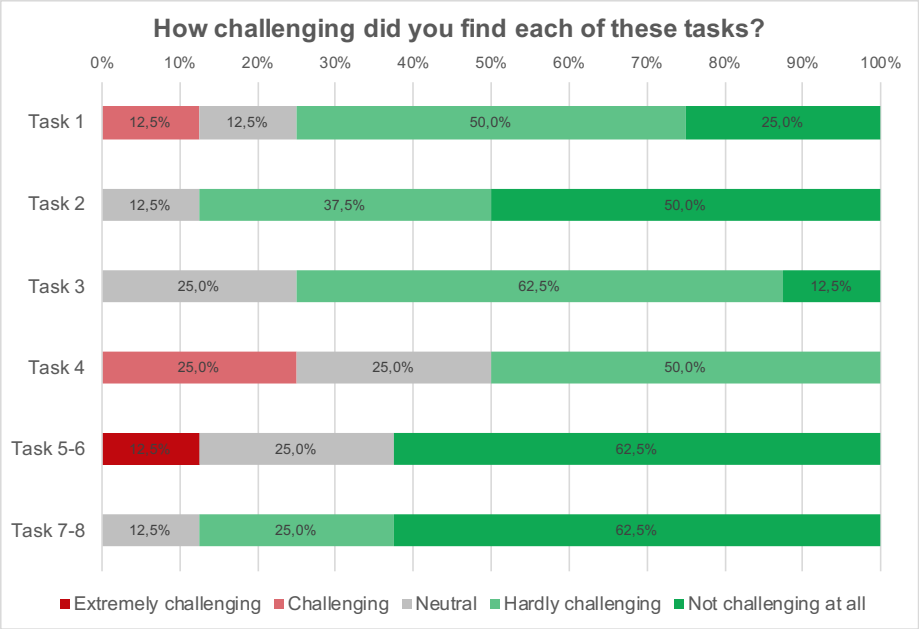


Figure 5.11: Likert responses for Perceived Level of Challenge (PC1; 8 participants)

Task 5 and 6 in combination were also regarded as challenging. While these tasks did not lead to many reported issues, participants argued that even a single manual change can have effects on the constraints that are difficult to correct, especially if it is not simply possible to switch one student with another, but multiple changes need to be made.
The remaining tasks were considered to be relatively less challenging, but three out of eight participants stated that the problem would become more complex if they had to take project requirements into account as well, which were excluded from the study.

Results: Satisfaction of Participants with TEASE (RQ9)

To investigate **RQ9**, we examined participants' perceived usefulness and perceived ease of use regarding TEASE, the results of which are shown in Figure 5.12 and Figure 5.13.

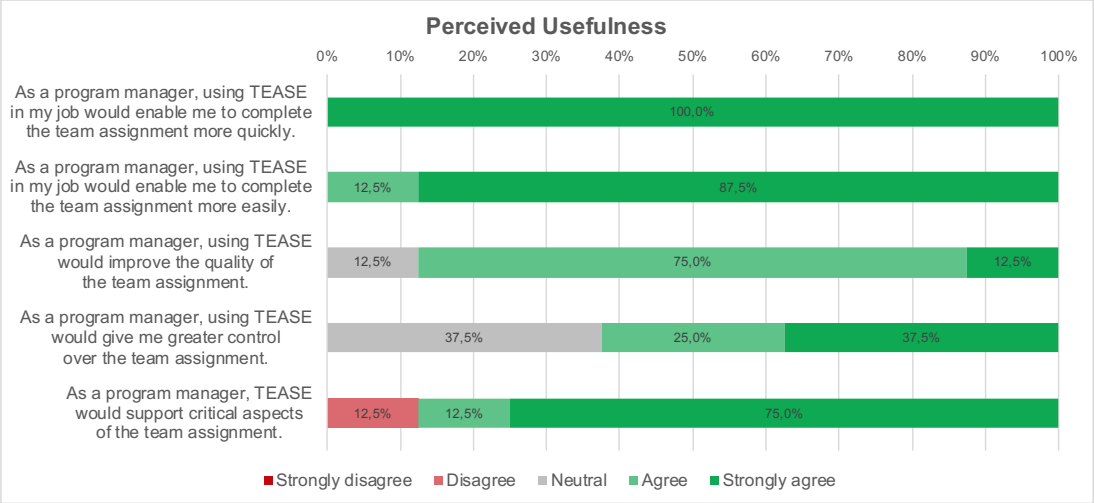


Figure 5.12: Likert responses for Perceived Usefulness (PU; 8 participants)

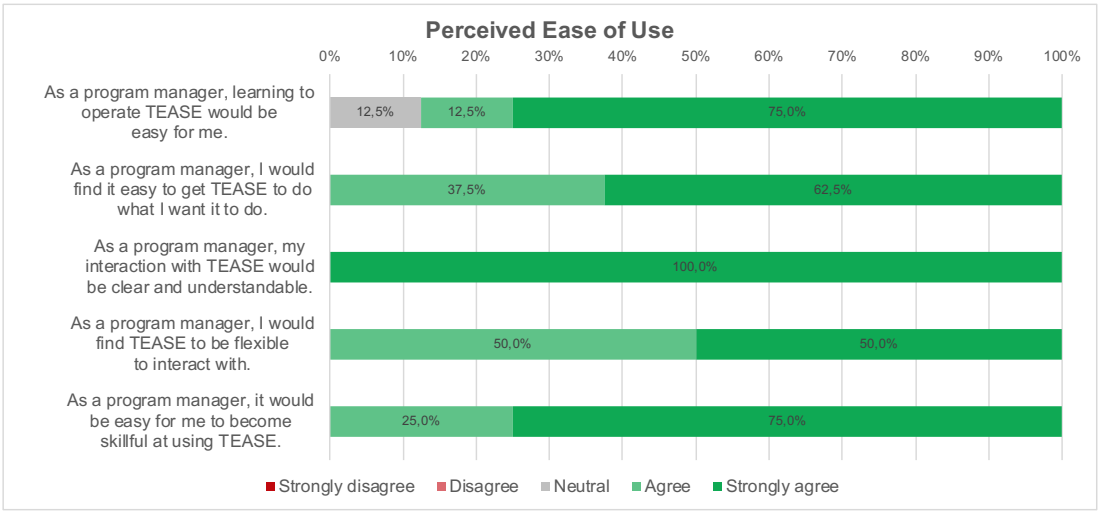


Figure 5.13: Likert responses for Perceived Ease of Use (PEOU; 8 participants)

The open-ended questions *ADV* and *DISADV* were coded using a provisional coding technique, and the resulting codes and the number of their occurrences are available in Appendix D.6. Participants referred mostly to the ability of TEASE to show a large amount of information in a condensed and easily understandable form, as well as the ability of the algorithm to take many criteria into account while they would fear to forget something with a manual assignment. Multiple participants also stated that

they felt TEASE saves a lot of time by providing an initial suggestion which can then be adapted.
In terms of disadvantages, three out of eight participants said they could not find any, while the rest stated that they were worried of not seeing the need for a manual adaptation because they trust the suggestion too much and do not reflect on it critically enough. Moreover, they negatively remarked that qualitative criteria and project requirements were not taken into account by the algorithm.
In addition to this data, participants were asked to state how much they trust TEASE to support them during team initialization, and whether they would recommend the tool to fellow educators. The results of this *RETR* question are shown in Figure 5.14. Thus, we obtained support for both **H9.1** abd **H9.2** in the sample of project leaders.

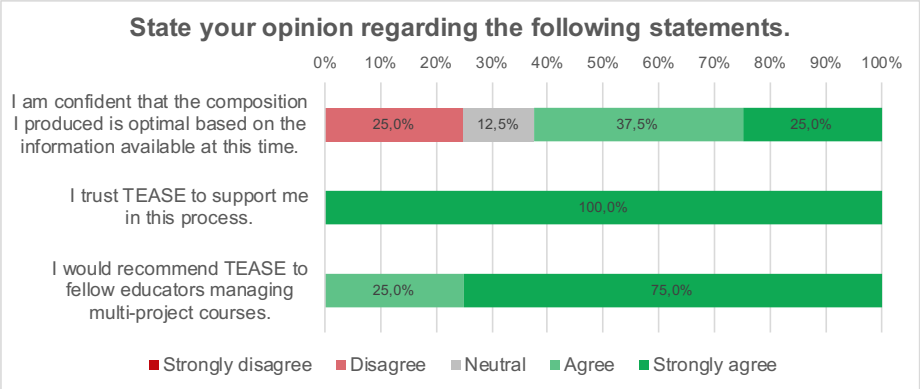


Figure 5.14: Likert responses for Recommendation & Trust (RETR; 8 participants)

Discussion & Implications

In the following, we discuss the results with regard to the research questions and suggest directions for future work.
RQ7 investigates whether new users can successfully perform the team initialization with minimal prior training. All eight participants were able to create an assignment and completed the majority of tasks without needing assistance, but the quality of the assignment configuration, measured by the number of constraints in the acceptable range, varied strongly. This implies that being able to set optimal constraints takes practice and one gets better when performing multiple initializations and seeing the effect of the configuration on the suggestions. The participants who did not set optimal constraints were able to see and reflect on the weaknesses of their assignment in Task 8: for instance, both participants who set the team size minimum to five noticed that some teams were too small and tied this to the fact that they set the constraint accordingly. We suspect that further training or practice would lead to a quick learning curve, but further research is needed to evaluate extended learnability.

With **RQ8** we examined what issues the participants have with the tool and how challenging they find the task they perform. Overall, the participants did not find the tasks very challenging. Interestingly, the tasks with the most variance in how the participants solved them (Tasks 2-4) also had the highest number of reported issues and were perceived as most challenging by the participants. In terms of the kinds of issues reported, the categories *awareness of functionality* and *locating functionality*, in sum 25% of the total issues, can easily be avoided by providing more extensive instructions or a more detailed introduction into the tool. The other two categories, *understanding task flow* and *understanding functionality* should also be investigated in a study examining extended learnability of the tool.
Finally, with **RQ9** we were interested in whether the participants are satisfied with TEASE as a solution for performing the team initialization. Both perceived usefulness and perceived ease-of-use scores were high among all participants. While the Technology Acceptance Model argues that there is a correlation between these two metrics and users' acceptance of a technology, it does not give indications as to a threshold above which users are likely to accept a tool [Dav89]. The responses indicate a strongly positive attitude of participants toward TEASE. Moreover, both of the hypotheses of this research question were supported by the results, and the coded open-ended comments of the participants lend further support to the results. The question here, however, is to what extent users are able to perceive the full usefulness of TEASE after performing a reduced set of team initializations tasks only once.
Overall, the results of the observational study show that users that were not directly involved in the development of TEASE like the program managers, and who have no prior experience with team initialization, can not only use the tool, but also achieve qualitatively good results with it. While the project leaders' positive feedback is encouraging regarding the future development and usage of TEASE, further research is needed to prove its applicability and usefulness with other user groups and in completely different contexts, also outside of the educational domain.

5.5 Limitations & Threats to Validity

This section discusses the limitations and threats to validity to the embedded case study. As it is recommended to address threats to validity for case studies in a systematic way [PSE04; Woh+12], we use the classification by Yin [Yin17], which consists of the following types of validity:

Construct validity concerns the question whether the study is measuring what the experimenter intended to measure, i.e., whether the treatment operationalizes the theory behind it.

Internal validity is threatened if there is a possibility that the observed outcome does not result from the treatment, for instance because there are other influencing factors.

External validity describes the generalizability of the findings to other cases beyond the investigated one.

Conclusion validity is the degree to which conclusions made on the basis of evaluation data are reasonable and valid.

While some threats to validity are unique to the research method used in one of the treatments, most apply to the whole case study. Thus, the limitations of all treatments are discussed at once, structured by the classification described above.

Construct Validity

Cook et al. [CCP90] list several factors that can limit construct validity, amongst others the *mono-method bias*, i.e., the usage of a single type of measure or observation leading to an inaccurate result. We mitigated this threat in the design of the case study by involving multiple methods, namely an experiment, a focus group workshop, and an observational study with an added questionnaire. Since the evaluation contains only one case, the iPraktikum, a *mono-operation bias* remains, even if the kinds of treatments varied.
There are also several other, social factors that can influence construct validity of the treatments that involved human experimenters and subjects. First, the introduction of the treatment itself might have made the subjects more sensitive to the effect of the treatment: for instance, the program managers during the lab validation might have been more aware of the time savings using TEASE and thus more conscious of not wasting time during team initialization, making the effect stronger. We addressed this

threat by using a focus group of program managers, expecting them to pay attention to possible biases of their peers.
The observational study with project leaders is possibly affected by the two effects *hypothesis guessing* and *evaluation apprehension* [CCP90]: there is a risk of subjects guessing or knowing the hypotheses and thus trying to act accordingly, or in general perform better than they would in an unobserved environment. To limit this threat, the hypotheses of the study were not shared with the subjects before their participation, even if they asked directly, and all subjects were reassured that their performance was not being measured, but the goal was to collect qualitative data on the usage of TEASE as well as new requirements.
Finally, a threat of *experimenter expectancies* influencing the results of the study applies to all treatments. Since we used action research as opposed to a purely observational approach in most validation steps, the experimenters were allowed to intervene and influence the process of the treatment. This threat was addressed by documenting all interventions and checking them with impartial peers. In this case, the iterative improvement of TEASE was the most important goal, thus we accepted the tradeoff of inadvertently impacting the results.

Internal Validity

The *absence of a control group* limits the internal validity of both dynamic validations: there is no direct comparison to what the results would have been using the same instance of the course without introducing TEASE. However, the structure and processes of the iPraktikum are well-defined and do not vary significantly between semesters, which enables us to compare the team initialization across course instances without fundamentally compromising the quality of the results.
The lab validation focused on the comparison of the team initialization process with and without TEASE using the same participant and project data. In this case, there is a possible effect of *maturation* because the program managers were already familiar with the students and projects in the course. We expected this effect to manifest itself especially when assigning Instructor Ratings, hence we chose not to compare the resulting ratings, but use the originally assigned ones for further steps of the process. Moreover, since effect of the treatment on the duration this process step was negligible, one can argue that the maturation effect is not fundamental in the rest of the lab validation either.
Finally, a possible threat to validity in all treatments involving material or instructions given to subjects is due to *instrumentation*, i.e., this material or instructions influencing subjects' intentions and actions in a way that impacts the results. Thus, questions and issues regarding the instructions themselves were excluded from the analysis in

the observational study and encouraged subjects to ask clarifying questions whenever they felt that something was unclear or ambiguous.

External Validity

The evaluation of TEMPO has produced promising results in the context of the iPraktikum, and the fact that the results of several types of treatments showed similar results further supports the hypotheses [RH09a]. However, it is questionable whether these results are generalizable to other educational or industry contexts.
Regarding the usage in other project courses that use different criteria for team initialization, TEASE was designed with extensibility and customizability in mind and would expect other instructors to be able to tailor the tool, just as we have iteratively tailored it during dynamic validation. Nevertheless, since TEASE has not yet been evaluated in another setting, it is unclear how the tool would perform with a larger number of students or differently configured objectives and constraints, for instance.
When it comes to an evaluation in industry, we opted against this setting for the scope of this thesis due to lengthy reviews and complex privacy and other concerns in especially large organizations that the interview study revealed [DB18b]. Moreover, we preferred a setting that is both fully available to study and under the control of the experimenters so that elements of action research can be introduced into some of the treatments. The amount of control over the course also made it possible to investigate further in case the data was unclear, e.g. by asking the project leaders or team coaches to clarify what was going on in their team.
Project courses present the students with a setting that is close to industry, typically based on teamwork and including activities like requirements analysis, system design, implementation and delivery [KABW14; SLPK15]. Instructors of such courses face many of the challenges of practitioners, including having to balance heterogeneous prior experience, cultural factors, or work habits and coping with incomplete, often self-reported, information about the participants [PSCB00]. Thus, we are optimistic that although the evaluation in an industrial setting is subject to future work, the results of the usage of TEASE in the setting of the iPraktikum are somewhat generalizable to applying TEMPO in a project-based organization in industry.

Conclusion Validity

Small sample size is a factor that limits conclusion validity for all studies: Lab validation was performed in three instances of the course with 33 teams in total, while static and dynamic validation I were conducted with only three program managers, and dynamic validation II involved eight project leaders. Wohlin et al. [Woh+12] recognize that in applied research, the typically small sample size limits conclusion validity, and prioritize the other types higher. At this early stage of the development of TEASE, this threat cannot be eliminated because a large user base is not available. The evaluation results nevertheless serve as a first indication of the tool's usefulness.
Regarding the observational study, using the Technology Acceptance Model to infer subjects' acceptance has been criticized since it relies purely on self-reported data instead of real actual system use [Chu09]. This threat was reduced by combining self-reported data with information about the resulting assignment configuration as well as number of issues that occurred, both of which describe actual system use.
Finally, the evaluation of TEASE took place in the setting of one multi-project course, the iPraktikum. While the setting is close to industry, the tool has not been evaluated in an industrial context. Thus, one cannot make empirically sound conclusions about the applicability and usefulness of TEASE or TEMPO in industry. Further research is needed to investigate this, and we hope that we can conduct it in the future.

Chapter 6

Conclusion

> The most exciting breakthroughs of the twenty-first century will not occur because of technology, but because of an expanding concept of what it means to be human.
>
> John Naisbitt

This chapter summarizes the contributions of this dissertation and discusses possible directions for future work.

6.1 Contributions

The main goal of this dissertation was to create a framework for team composition in project-based organizations that increases the manageability of the team composition process by algorithmically supporting the staffing of project teams. This section summarizes the main contributions of this dissertation.

Analysis of Team Composition in Practice

We conducted empirical interview study with practitioners in project-based organizations in the software industry and analyzed their processes and tools for team composition. The result was a categorization of team composition processes in industry with the three observed process types *Decentralized Process*, *Centralized Process*, and *Networked Process*, as well as a formalization of process states and roles involved in team composition. We also assessed current approaches to algorithmic team composition by synthesizing related work and reviewed team factors and group development theory models. The results showed a lack of tool support for team composition that is

neither fully algorithmic nor fully manual and goes beyond the search for employees with specific skills.

Framework for Team Composition in Project-based Organizations

We developed TEMPO, a framework for team composition in organizations that is tailorable to pure project-based organizations, project team organizations and matrix organizations. TEMPO provides algorithmically optimal suggestions based on specified constraints and objectives which can then be adapted based on human knowledge and experience with the people involved. The framework offers the workflows *Project Preparation*, *Team Initialization*, *Team Composition Monitoring*, and *Employee Information*. TEMPO's aim is to increase the manageability of the team composition process.

Instantiation and Empirical Evaluation of TEMPO

We instantiated the two TEMPO workflows *Project Preparation* and *Team Initialization* in the context of a large multi-project capstone course using flat staffing. We developed TEASE, a reference implementation tailored of the previously manual team composition process and criteria of this course.
TEASE was applied in an embedded case study using qualitative and quantitative research methods: A quasi-experiment provided first proof of its applicability in this setting and showed that TEASE shortens the duration of the team initialization by 61% while resulting in comparable or better assignment quality. In the subsequent focus group workshops, we employed action research to iteratively improve TEASE. An observational study showed that domain experts who have never used the tool before can successfully perform the team initialization with minimal prior training. Through this case study, we demonstrated the applicability of TEMPO in the iPraktikum as an example of a project-based organization.

6.2 Future Work

We have identified several aspects of TEMPO and TEASE that offer room for further research or development work.

Further Research for TEMPO

We envision multiple areas in which TEMPO can be extended. As of now, only optimization was considered for generating team suggestions. In the future, machine learning or deep learning approaches and algorithms for classification or regression can be evaluated regarding their usefulness within the framework, especially when there is a large amount of data available for training, as briefly discussed in Section 3.7. Machine learning approaches can also be useful for classifying textual data such as employee CVs or comments [IBM16].
TEMPO has been designed to be applicable to organizational types that range from pure project-based organizations to matrix organizations with a focus on the software engineering industry. Another possible area of future work is to extend the framework to other types of organizations or other industries where project teams are initialized from a larger pool of people, for instance for the less permanent role system of a Holacracy.
In terms of team composition criteria, we consider the addition of personality factors an interesting future direction. Further research has to be conducted on how to model personality and which personality configurations can or should be taken into account during team composition. There are tools on the market for personality analysis in teams[1] which could be studied regarding their integratability into TEMPO.

Extension and Improvement of TEASE

TEASE is an instantiation of two of TEMPO's workflows in the context of the multi-project course iPraktikum. Staying in this context, we suggest to extend the tool by supporting more complex criteria such as personality. TEASE could also be extended to cover more of TEMPO's workflows, for instance the *Team Composition Management* workflow which has not been instantiated yet: Some of our previous work examined the use of metrics in project courses [ADB16] and could be used as a starting point. Another possible extension would be to adapt the architecture of the tool to allow distributed access to the team composition, for instance to enable multiple program managers to collaborate across locations.

[1] e.g. `https://www.saberr.com/base`

Going beyond the context of the iPraktikum, TEASE could be further developed to be tailorable to other multi-project courses by enabling the dynamic configuration of criteria and supporting a flexible data format.
Another direction for the improvement of TEASE is to include alternative suggestion algorithms, notably multiobjective optimization approaches. This would enable the program managers to convert some constraints into objectives and thus e.g. make it easier to achieve a skill balance in the teams. Moreover, this would make it possible to factor in project requirements, which are not yet taken into account in TEASE's algorithmic suggestions. Especially iterative solution approaches such as the NIMBUS method described in Section 3.7 seemed promising and could be compared to the existing single-objective algorithm in terms of assignment quality and performance. Moreover, the addition of suggestion algorithms that support fuzzy data would make the tool more robust toward missing or ambiguous student information.

Further Evaluation of TEMPO and TEASE

In this dissertation, we evaluated TEMPO by applying its reference implementation TEASE to the context of a multi-project course. The evaluation of the framework in industry was not possible within the scope of this thesis due to a resistance in the interviewed organizations to release employee and project data as well as a reluctance to closely study the impact of the tool on the team initialization process. Future research should aim to prove the applicability of TEMPO in project-based organizations in the software engineering industry.
Another possible focus for future work is to examine whether algorithmically supporting the team composition process leads to qualitatively better teams. Ultimately, an algorithm can only optimize for quantifiable goals, and the underlying constraints and objectives are a major determinant of the resulting assignment. Basing the team composition on pre-defined, carefully selected criteria will increase the quality of the resulting teams, but team quality and team performance are influenced by many factors which cannot be examined in isolation. In the future, we hope to be able to conduct further work on which criteria influence the output and well-being of teams and use the results to improve both TEMPO and TEASE.

Appendices

Appendix A
Definitions

The following is an alphabetically sorted list of terms and definitions that are used in this dissertation.

Action Research Research strategy which is closely related to a case study, but "is focused on and involved in the change process" [RHRR12].

Analysis Object Model "The analysis object model describes the application domain concepts that the system manipulates and the user-visible interfaces of the system" [BD09].

Case Study "Case study in software engineering is an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified" [RH09a].

Cohesion The degree to which a group acts as a whole and its members feel a sense of commonality and belonging [Yuk13].

Diversity The degree of variation in attributes such as culture, gender, race, age, or education across team members [RJ15].

Motivation "The processes that account for an individual's intensity, direction, and persistence of effort toward attaining a goal" [RJ15].

Object Design Model "A detailed object model representing the application and solution objects that make up the system. The object design model includes detailed class specifications, contracts, types, signatures, and visibilities for all public operations" [BD09].

Organization "A consciously coordinated social unit, composed of two or more people, that functions on a relatively continuous basis to achieve a common goal

or set of goals. [...] By this definition, manufacturing and service firms are organizations, and so are schools, hospitals, churches, military units, retail stores, police departments, and local, state, and federal government agencies" [RJ15].

Organizational Structure "The way in which job tasks are formally divided, grouped and coordinated" [RJ15].

Project-based Organization (PBO) An organizational form "in which the project is the primary unit for production organisation, innovation, and competition" [Hob00]. Project-based organizations "involve the creation of temporary organizations or sub-units for the performance of project-based tasks. Upon completion of the project, the temporary organization or subunit is dissolved and its participants move on to new projects, involving new human capital arrangements" [DeF02].

Project Leader "A management role responsible for planning, monitoring, and controlling a single team" [BD09]. Synonyms: **Project Lead**, **Team Lead**, **Team Leader**.

Requirement "A function that the system must have (a functional requirement) or a user-visible constraint on the system (nonfunctional requirement)" [BD09].

Role "A set of expected behavior patterns attributed to someone occupying a given position in a social unit" [RJ15].

Software Engineering " describes the collection of techniques that apply an engineering approach to the construction and support of software products. [...] Whereas computer science provides the theoretical foundations for building software, software engineering focuses on implementing the software in a controlled and scientific way" [FB14].

Staffing Profile The evolution of a team's composition over time. Common approaches are **Flat Staffing**, i.e., estimating the number of people needed for completing the project and assigning them to the team from the very beginning, and **Gradual Staffing** which changes the team's composition according to the project's demand, for instance by approximating the manpower needed by using a Rayleigh curve [Mad07].

Team "A team is a small number of people with complementary skills who are committed to a common purpose, set of performance goals, and approach for which they hold themselves mutually accountable" [KS93].

Team Assignment The activity of assigning one or multiple persons to a team, making them a member of that team. Synonyms: **Team Allocation**, **Team Staffing**.

Team Composition (noun) The collection of characteristics of all members of that team. These characteristics can be analyzed as an aggregated profile, in terms of heterogeneity, or by looking at how individual traits impact the team [Bel07; Ste06].

Team Composition (activity) In the scope of this thesis, when this term is used as a verb ('to compose a team') it refers to a combination of the activities **Team Initialization and Team Assignment**.

Team Initialization The activity of allocating one or multiple persons to a new team, thus creating its initial composition.

Technology Transfer Model A methodology for the "successful transfer of knowledge and technology from research to practice" [GGLW06]. It involves "Using several assessments [...] (and several companies if possible) to find topics for research and formulate problem statements while studying the field and domain" [Woh+12] and suggests the steps lab validation, static validation, and dynamic validation to evaluate a candidate solution.

Tornado Model A process model that "combines the Unified Process with Scrum elements. The Tornado model focuses on scenario-based design starting with visionary scenarios funneling down to demo scenarios. [...] In addition to formal models used for analysis and design, Tornado encourages the developer to use informal models as communication medium" [BKW12].

Use case "A use case describes a function of the system in terms of a sequence of interactions between an actor and the system" [BD09].

Visionary Scenario "Scenario that describes a future system. Visionary scenarios are used both as a point in the modeling space by developers as they refine their ideas of the future system and as a communication medium to elicit requirements from users" [BD09].

Workflow "A thread of cohesive and mostly sequential activities performed by project participants that produce artifacts" [BD09].

Appendix B

iPraktikum Team Initialization Questionnaire

The following pages include the questionnaire sent to the students of the iPraktikum during the Swift Bootcamp (cf. Section 4.1).

Dear {TOKEN:FIRSTNAME},

With this survey we want to learn more about you and your existing knowledge, experiences and interests. We will use this information to assign you to one of the project teams. Don't worry, we will also take your project preferences into account: you will be able to tell us those right after the Kickoff meeting.

Please keep in mind that none of this information is used for assessing your contribution to the course in any way. You have already been selected for the iPraktikum and have successfully gone through the intro course - now we want to learn more about you to make sure you have the best possible experience.

You will receive a confirmation to your email address {TOKEN:EMAIL}. Please check your email after submitting this survey!

Section A: About you

These are some basic questions about you as a person, your studies and the devices you own.

A1. **What is your major (Studiengang)?**

- ☐ BSc Informatics
- ☐ BSc Information Systems
- ☐ BSc Games Engineering
- ☐ BSc TUM-BWL, Ingenieurwissenschaften
- ☐ BSc Mechanical Engineering, Electrical Engineering
- ☐ MSc Informatics, Biomedical Computing, Data Engineering
- ☐ MSc Information Systems
- ☐ MSc Games Engineering
- ☐ MSc TUM-BWL, Ingenieurwissenschaften
- ☐ MSc Mechanical Engineering, Electrical Engineering
- ☐ MSc Automotive Engineering
- ☐ MSc Robotics
- ☐ Other

Other

A2. **What is your semester number in your current major?**

If you are in a MSc degree, your BSc semesters do not count!

- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7
- ☐ 8
- ☐ 9
- ☐ 10

A3. **State your language skills.**

	Basic such as A-Level (A1/A2)	Medium such as B-Level (B1/B2)	High such as C-Level (C1/C2)	Native
German	☐	☐	☐	☐
English	☐	☐	☐	☐

A4. **Here are some personality trait pairs that may or may not apply to you. Please specify the extent to which you agree or disagree with each statement. You should rate the extent to which the pair of traits applies to you, even if one of them applies more strongly than the other.**

Note: There is no 'right' or 'wrong', 'better' or 'worse' way of answering these questions. Please give honest answers - it will help us create teams in which everyone has the best possible experience :-)

	Disagree strongly	Disagree moderately	Disagree a little	Neither agree nor disagree	Agree a little	Agree moderately	Agree strongly
I see myself as extraverted, enthusiastic.	☐	☐	☐	☐	☐	☐	☐
I see myself as critical, quarrelsome.	☐	☐	☐	☐	☐	☐	☐
I see myself as dependable, self-disciplined.	☐	☐	☐	☐	☐	☐	☐
I see myself as anxious, easily upset.	☐	☐	☐	☐	☐	☐	☐
I see myself as open to new experiences,	☐	☐	☐	☐	☐	☐	☐
I see myself as reserved, quiet.	☐	☐	☐	☐	☐	☐	☐
I see myself as sympathetic, warm.	☐	☐	☐	☐	☐	☐	☐
I see myself as disorganized, careless.	☐	☐	☐	☐	☐	☐	☐
I see myself as calm, emotionally stable.	☐	☐	☐	☐	☐	☐	☐
I see myself as conventional, uncreative.	☐	☐	☐	☐	☐	☐	☐

Section B: Your iOS Skills and Devices

Tell us about your experience in iOS development and devices you can use for development in the course.

B1. **What are your experiences with developing for the Apple platform?**

Choose the highest answer that applies to you.

- I have no experience in Apple platform development other than the intro course. ☐
- I was involved in the development of a native Apple application, but I had another role than developer (e.g. tester). ☐
- I have been an active developer for a native Apple application. ☐
- I have submitted my own native Apple application(s) to the AppStore. ☐

B2. **Please provide one or multiple links to the application(s) in the AppStore.**

B3. **Please describe your iOS development experience in more detail. Be specific: name Frameworks you worked with and what you did with them, and describe the projects you worked on and how long you were involved in them (provide links where possible). (max. 400 characters)**

B4. **I felt that the intro course was...**

not challenging at all	hardly challenging	medium challenging	very challenging	extremely challenging
☐	☐	☐	☐	☐

B5. **Do you own one of these Apple devices that you can use for development during this course?**

If you do not own one of these Apple devices, just leave this question empty. By the way, we know that you already told us this when applying for the iPraktikum, but we are asking again to make sure to get the most up-to-date information, and this time we also want to know the model of your device.

- ☐ iPhone - I own one of the following models: 5s, 6, 6 Plus.
- ☐ iPhone - I own one of the following models: SE, 6s, 6s Plus, 7, 7 Plus, 8, 8 Plus, X.
- ☐ iPad - I own one of the following models: mini 2, mini 3, mini 4, Air, Air 2.
- ☐ iPad - I own one of the following models: fifth-generation/2017, Pro.
- ☐ Mac or MacBook - I own a Mac that runs macOS Mojave.
- ☐ Apple Watch - I own an Apple Watch.

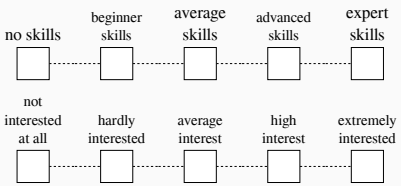
Section C: Other skills and interests

In this section we want to learn more about your skills and interests in a couple of software engineering and related fields. With this we can staff our project teams with the right set of skills and provide you with a project which fits your personal interests.

Each questions first asks about your current skill level and afterwards about your personal interest in the topic.

C1. **Please state your experience and interest in the topic of Frontend Development (e.g. Angular.JS, JQuery, HTML5/CSS3 etc.).**

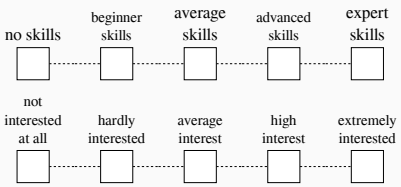
Please make two selections - one for your skills, and one for telling us how interested you are in learning more about the topic.



C2. **Provide us with examples of your advanced or expert knowledge in Web Frontend Development. Keep it short and be as specific as possible: name e.g. projects you worked on and the frameworks and technologies you worked with personally, a link to a github project etc. (max. 200 characters)**

C3. **Please state your experience and interest in the topic of Server-side Development (e.g. NodeJS, Django etc.).**

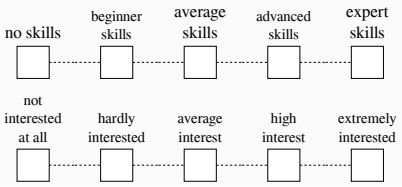
Please make two selections - one for your skills, and one for telling us how interested you are in learning more about the topic.



C4. **Provide us with examples of your advanced or expert knowledge in Server-side Development. Keep it short and be as specific as possible: name e.g. projects you worked on and the frameworks and technologies you worked with personally, a link to a github project etc. (max. 200 characters)**

C5. **Please state your experience and interest in the topic of Embedded Development (e.g. sensors, Arduino etc.).**

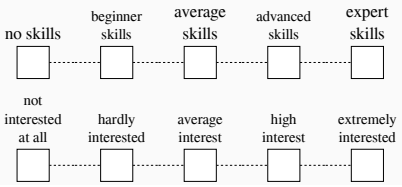
Please make two selections - one for your skills, and one for telling us how interested you are in learning more about the topic.



C6. **Provide us with examples of your advanced or expert knowledge in Embedded Development. Keep it short and be as specific as possible: name e.g. projects you worked on and the frameworks and technologies you worked with personally, a link to a github project etc. (max. 200 characters)**

C7. **Please state your experience and interest in the topic of Virtual and Augmented Reality.**

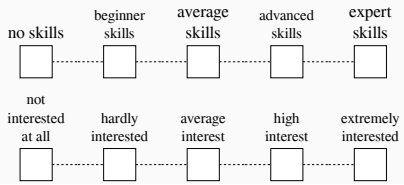
Please make two selections - one for your skills, and one for telling us how interested you are in learning more about the topic.



C8. **Provide us with examples of your advanced or expert knowledge in Virtual and Augmented Reality. Keep it short and be as specific as possible: name e.g. projects you worked on and the frameworks and technologies you worked with personally, a link to a github project etc. (max. 200 characters)**

C9. **Please state your experience and interest in the topic of Machine Learning and Algorithms (e.g. you have worked with TensorFlow, Blockchain, classification or image/voice recognition algorithms etc.).**

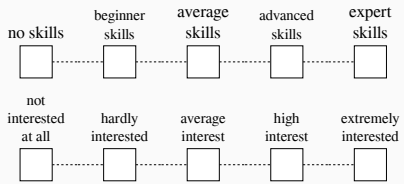
Please make two selections - one for your skills, and one for telling us how interested you are in learning more about the topic.



C10. **Provide us with examples of your advanced or expert knowledge in Virtual and Machine Learning and Algorithms. Keep it short and be as specific as possible: name e.g. projects you worked on and the frameworks and technologies you worked with personally, a link to a github project etc. (max. 200 characters)**

C11. **Please state your experience and interest in the topic of UI/UX Design (e.g. good Photoshop skills etc.).**

Please make two selections - one for your skills, and one for telling us how interested you are in learning more about the topic.



C12. **Provide us with examples of your advanced or expert knowledge in Virtual and UI/UX Design. Keep it short and be as specific as possible: name e.g. projects you worked on and the frameworks and technologies you worked with personally, a link to a github project etc. (max. 200 characters)**

C13. **Do you have other skills or interests which would be a good contribution to your iPraktikum team? Tell us what we forgot to ask. (max. 200 characters)**

Appendix C
Lab Validation

C.1 Full Result Tables

The below tables show the results of the lab validation using TEASE in three instances of the iPraktikum described in Section 5.1. The teams are numbered in reversed chronological order: T01-T11 were part of the winder semester 2016/17 instance of the course, T12-T23 are from the summer semester 2016, and T23-T33 belong to the winter semester 2015/16 instance.

Table C.1: Assignment quality, manual vs. TEASE-supported process; T01-T09

Team (anonymized)	**T01**	**T02**	**T03**	**T04**	**T05**	**T06**	**T07**	**T08**	**T09**
Avg. priority in course	5.00	6.30	6.00	4.10	5.50	9.40	6.00	5.30	4.20
Avg. priority in team - manual	1.38	1.57	1.14	1.13	1.71	4.00	1.30	2.00	1.00
Avg. priority in team - TEASE	1.25	1.57	1.00	1.00	1.71	3.86	1.43	1.29	1.13
Broken constraints - manual	1	1	0	0	0	0	1	0	0
Broken constraints - TEASE	0	0	0	0	0	0	0	0	0

Table C.2: Assignment quality, manual vs. TEASE-supported process; T10-T18

Team (anonymized)	**T10**	**T11**	**T12**	**T13**	**T14**	**T15**	**T16**	**T17**	**T18**
Avg. priority in course	8.50	5.80	7.08	5.49	6.55	8.97	8.22	5.84	3.77
Avg. priority in team - manual	4.57	1.14	2.67	2.50	2.33	2.83	2.33	1.00	1.50
Avg. priority in team - TEASE	4.14	1.13	2.67	1.50	1.33	2.17	2.00	1.00	1.29
Broken constraints - manual	2	0	3	1	0	0	1	0	0
Broken constraints - TEASE	0	0	0	0	0	0	0	0	0

Table C.3: Assignment quality, manual vs. TEASE-supported process; T19-T27

Team (anonymized)	T19	T20	T21	T22	T23	T24	T25	T26	T27
Avg. priority in course	8.23	4.27	6.31	6.19	7.06	6.03	3.67	5.71	5.37
Avg. priority in team - manual	3.17	1.33	1.67	2.55	2.33	2.33	2.00	2.29	2.71
Avg. priority in team - TEASE	2.50	1.00	1.83	2.50	2.50	2.00	1.29	2.14	1.57
Broken constraints - manual	0	0	0	1	1	2	1	0	0
Broken constraints - TEASE	0	0	0	0	0	0	0	0	0

Table C.4: Assignment quality, manual vs. TEASE-supported process; T28-T33

Team (anonymized)	T28	T29	T30	T31	T32	T33
Avg. priority in course	5.51	7.04	8.19	2.51	4.82	6.15
Avg. priority in team - manual	2.75	3.43	4.00	1.00	2.43	2.00
Avg. priority in team - TEASE	2.00	2.43	2.43	1.00	1.71	1.57
Broken constraints - manual	1	0	0	0	0	0
Broken constraints - TEASE	0	0	0	0	0	0

C.2 Graphical Representation of Results

The below figures show a graphical representation of the objective function value of the 33 teams in three iPraktikum instances during the lab validation described in Section 5.1. We include a box plot of the objective function value, and larger images of the spider chart distributions previously shown in Figure 5.4.

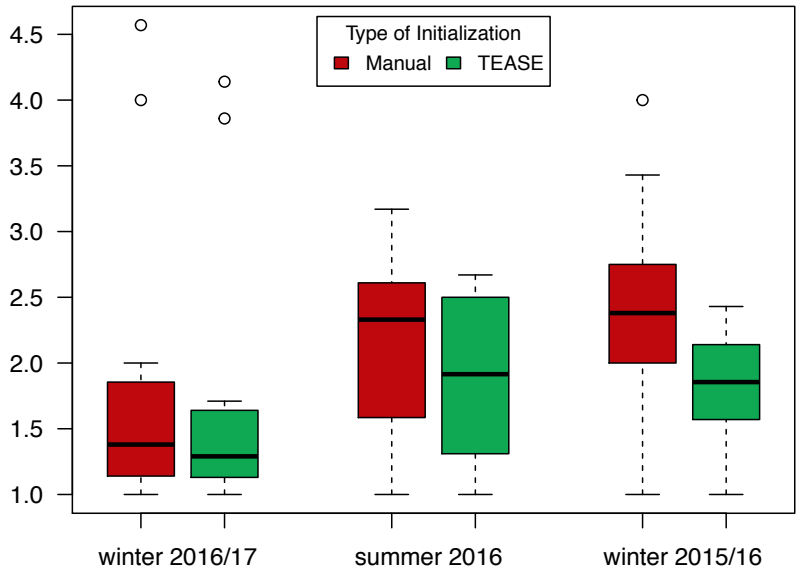
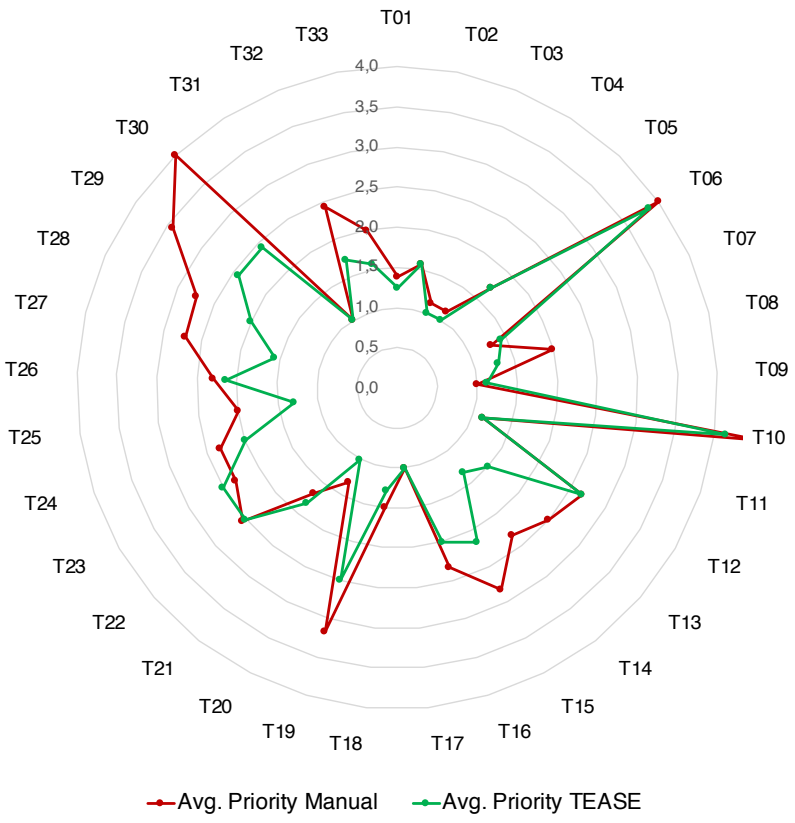
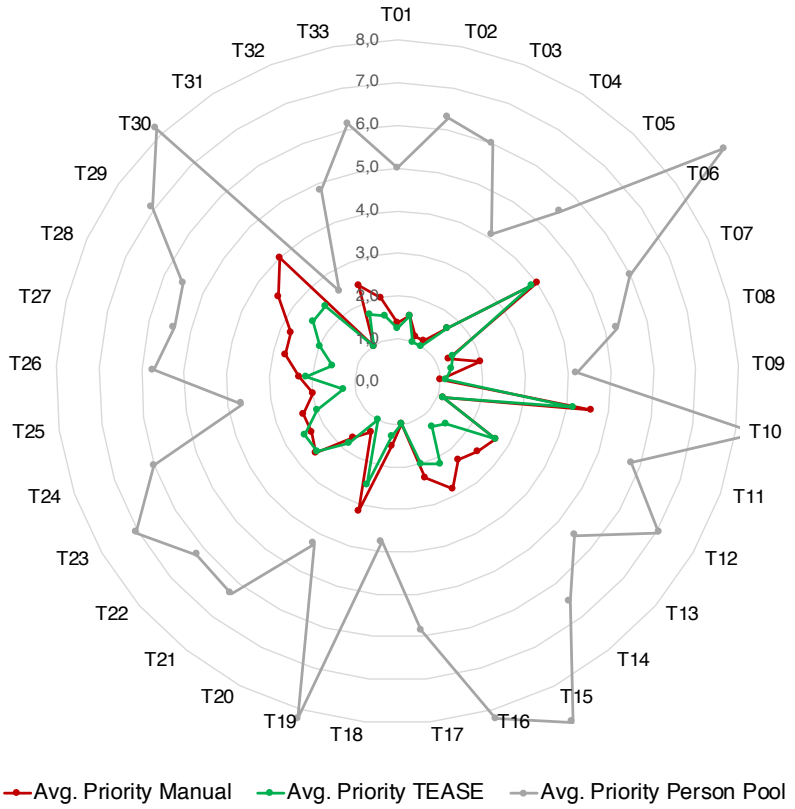


Figure C.1: Box plot of the objective function value $F_M(t)$ and $F_T(t)$ in three instances of the iPraktikum



(a) Assignment only



(b) Including avg. priority overall

Figure C.2: Comparison of manual and TEASE-supported team initialization over three semesters in reversed chronological order (full size)

Appendix D

Dynamic Validation

The following pages contain the material from the observational study described in Section 5.4. We include the informed consent form signed by participants, a one-page introduction, a more detailed set of task descriptions, as well as the questionnaire that participants answered immediately after finishing the tasks using TEASE.

D.1 Informed Consent Form

Think-aloud study TEASE
Informed Consent Form

Goals of the study

This research aims to evaluate the usefulness of and attitude toward TEASE, a tool for team allocation in software engineering project courses, from the point of view of a domain expert who has not used the tool before.

Subject's Understanding

- I agree to participate in this study and understand that my participation is voluntary.
- I understand that all data collected will be limited to this use or other research-related usage at the Chair for Applied Software Engineering at TUM.
- I understand that I will not be identified by name in any final product of the study and that there will be no possibility to infer the details of my person in any publication of the data.
- This study involves the audio and screen recording of your usage of the tool as well as your think-aloud comments and conversation with the experimenter. The recordings will be kept safe and confidential for archiving purposes, and they will only be listened to by the researchers involved in the study.
- I understand that transcripts of what I say during the study may be reproduced in whole or in part in presentations or written products that result from this study, but neither my name nor any other identifying information will be used alongside them.
- I understand that I may withdraw from the study at any time with no adverse consequences.
- I acknowledge that the contact information of the researcher and their advisor have been made available to me along with a duplicate copy of this consent form.

By signing below I acknowledge that I have read and understood the above information.

_____

Full Name

_____ Garching b. München, ____.____.2019

Signature

Technische Universität München
Faculty of Informatics
Chair for Applied Software Engineering
Boltzmannstr. 3
85748 Garching b. München

Dora Dzvonyar, M.Sc.
Tel: +49 (0)89 28918228
Mobil: +49 (0)176 20316218
dora.dzvonyar@tum.de
www1.in.tum.de

D.2 Study Introduction

Introduction to the user study
TEASE: Team Initialization in the iPraktikum

This research aims to evaluate TEASE, a tool for team allocation in software engineering project courses. You have been selected for this study based on your experience as a project lead in the iPraktikum. In this study, you will take on the role of program manager and perform the assignment of students to teams.

In the iPraktikum, we compose the teams based on the following criteria:

- **High project priority:** We want students to be assigned one of their top choices of project to ensure that they are highly motivated and prevent drop-outs.
- **Team size:** Most of the time, we want teams to have comparable size. Sometimes there is a different requirement for a single project.
- **Balance of skills:** We want less experienced students to learn from their more experienced peers, so we distribute the experienced students over all teams and aim for a balance of experience on each team.
- **Development and test devices:** We want each team to have around the same number of test and development devices (iPhones/iPads and Macs).
- **Female students:** Single-gender teams have shown to not perform as well as mixed-gender ones, to we want at least one female student per team.
- **Experience and/or interest in specific technology:** Some projects have specific topics or technologies (e.g. Machine Learning) at their core. We want these teams to have as many students as possible who either have experience, or are very motivated to learn more in this area.

During the course of this study, you will compose teams for a fictitious instance of the course using the criteria above. You will use TEASE, a tool that has been developed to algorithmically assist you in this process. You will do the following tasks while thinking aloud in the process:

1. **Rate students' experience:** Based on self-reported and tutor-reported information, you will divide students into 4 levels of experience.
2. **Set constraints:** Using the data in the pool of students, you will decide on appropriate constraints to set for assigning students to teams.
3. **Generate and evaluate suggestions:** TEASE will generate suggestions for assignments for you, which you will inspect and evaluate. An assignment is a mapping of each student in the course to exactly one project team.
4. **Adapt suggestions:** You will adapt the given suggestions using requirements given to you and see how these affect the assignment criteria. Sometimes it's acceptable to break constraints (judge this by your personal experience)

In the end, you will fill out a questionnaire to collect your opinion and feedback on the tool.

D.3 Task Descriptions

Page 1

User Study Tasks
TEASE: Team Initialization in the iPraktikum

*You will complete a series of tasks using TEASE. During the whole study, try to verbalize all of your **actions, intentions and thoughts**. If a task is unclear, indicate to the experimenter that **you need clarification and read aloud the instruction that is unclear.***

*If you encounter a problem, try to solve it first, as if you were by yourself, but do not use any resources other than the current tool. If you can't solve the problem by yourself, **indicate to the experimenter that you need help and ask them a specific question.***

Please read the instructions for a section (e.g. section A) completely before starting to solve it.

Overview & Context

You assume the role of program manager of the project course iPraktikum. Your task is to distribute the students in the course to the 12 projects. For this, you have self-reported and tutor-reported data about each student. Use the criteria on the introduction sheet as basis for your decisions.

A: Rate students' experience

You see all students that have signed up for the course in the Person Pool. In order to create a comparable level of experience, your first task is to merge the self-reported and tutor-reported data into one Instructor Rating (Novice - Intermediate - Advanced - Expert). All but 5 students have already been rated.

1. Inspect the 5 unrated students and rate each of them. Go by self-reported skills and also the tutor's rating of the student. You can use the already rated students as a reference.

*Note: The Intermediate rating should be your baseline. The Expert rating should only be given to students who have experience in iOS development or are extremely strong in programming ability. **Disregard their interest levels completely for now.***

B: Set constraints

As a next step, you will configure TEASE to make suggestions for you. The objective is always the same: minimize the project priority on the team. You can set the constraints yourself.

2. Open the statistics of the Person Pool and derive the maximum constraints that you can set globally for all teams for the following criteria:

   a) team size range (distributing the students equally to the 12 teams),
   b) minimum number of female students,

c) minimum number of development devices (MacBooks), and
d) minimum number of test devices (iPhones/iPads) per team.

Set these constraints in the constraint view as global constraints for all teams.

3. In order to create a balance of skills on the team, you will create constraints to distribute the different levels of instructor rating equally over teams. Look at the number of students with each instructor rating and create constraints **for 3 out of the 4 levels**.

C: Generate and evaluate suggestions

The Assignment Problem is all set up in TEASE and you are ready to generate suggestions. You will also review the quality of the suggestion.

4. Generate a suggestion with TEASE. Open the statistics for all teams and reflect on the quality of the assignment. Have the criteria been met? Is there room for improvement? No action required, just share your thoughts as they come.

Note: You can sort the teams to better see the balance of skills on each team.

D: Adapt suggestions

There are some requirements for this instance of the course where you have to make manual adaptations to the assignment. TEASE will show you how these influence the previously set criteria.

5. The customer of the SIMO project has requested the student Jesus Velasco to be on their team because he has worked with a neighboring department and thus has a lot of domain knowledge. Move this student to the team.

6. Evaluate what changed with regard to the criteria. If constraints were broken, reflect on whether this is acceptable for you as program manager. Pick the broken constraint that seems most important to you and fix it.

7. The GOAL team needs an additional student who is extremely interested in Machine Learning to learn from the experienced ML fans on the team. Use the highlighting feature to find students with an interest level of “high” or “very high” in Machine Learning. The students should have a minimum instructor rating of “intermediate”.

8. If you find a student you can move into the GOAL team without decreasing their project priority by more than 3, perform the change. Evaluate what changed with regard to the criteria. If constraints are broken, reflect on whether this is acceptable to you as a program manager (you don’t need to fix them).

Congrats — you’re done! Please fill out the post-questionnaire. Thank you for supporting this research!

D.4 Questionnaire

TEASE User Study Post-Questionnaire

Please answer the following questions from the perspective of the role you took on during the study: a program manager of the iPraktikum.

* Required

1. **How challenging did you find each of these tasks?** *

Mark only one oval per row.

	Not challenging at all	Hardly challenging	Neutral	Challenging	Extremely Challenging
Task 1: Rating students' experience	◯	◯	◯	◯	◯
Task 2: Setting constraints for team size, devices, female students	◯	◯	◯	◯	◯
Task 3: Setting constraints for skill balance	◯	◯	◯	◯	◯
Task 4: Evaluating generated suggestions	◯	◯	◯	◯	◯
Task 5-6: Assigning a specific student to a team	◯	◯	◯	◯	◯
Task 7-8: Locating students with specific skills and interests	◯	◯	◯	◯	◯

2. **For each task that you rated "challenging" or "extremely challenging", please describe why you chose this answer.** *

3. **Please state your opinion regarding the following statements.** *

Mark only one oval per row.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I am confident that the composition I produced is optimal based on the information available at this time.	◯	◯	◯	◯	◯
I trust TEASE to support me in this process.	◯	◯	◯	◯	◯
I would recommend TEASE to fellow educators managing multi-project courses.	◯	◯	◯	◯	◯

4. **What do you think are advantages of using TEASE compared to a purely manual process?** *

5. **What do you think are disadvantages of using TEASE compared to a purely manual process?** *

6. **Please state your opinion regarding the following statements.** *
   Mark only one oval per row.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
As a program manager, using TEASE in my job would enable me to complete the team assignment more quickly.	◯	◯	◯	◯	◯
As a program manager, using TEASE in my job would enable me to complete the team assignment more easily.	◯	◯	◯	◯	◯
As a program manager, using TEASE would improve the quality of the team assignment.	◯	◯	◯	◯	◯
As a program manager, using TEASE would give me greater control over the team assignment.	◯	◯	◯	◯	◯
As a program manager, TEASE would support critical aspects of the team assignment.	◯	◯	◯	◯	◯

7. **Please state your opinion regarding the following statements.** *
Mark only one oval per row.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
As a program manager, learning to operate TEASE would be easy for me.	◯	◯	◯	◯	◯
As a program manager, I would find it easy to get TEASE to do what I want it to do.	◯	◯	◯	◯	◯
As a program manager, my interaction with TEASE would be clear and understandable	◯	◯	◯	◯	◯
As a program manager, I would find TEASE to be flexible to interact with.	◯	◯	◯	◯	◯
As a program manager, it would be easy for me to become skillful at using TEASE.	◯	◯	◯	◯	◯

8. **Is there anything that we should know but didn't ask?**

D.5 Results: Assignment Problem Configuration

The following table shows the assignment problem configurations of all participants in the observational study described in Section 5.4.

Participant	**P01**		**P02**		**P03**		**P04**		**P05**		**P06**		**P07**		**P08**	
Constraint c	c_{min}	c_{max}	c_{min}	c_{max}	c_{min}	c_{max}	c_{min}	c_{max}	c_{min}	c_{max}	c_{min}	c_{max}	c_{min}	c_{max}	c_{min}	c_{max}
Team Size	6	8	6	7	6	7	6	7	6	7	5	7	6	7	5	8
Expert	0	1	–	–	0	1	–	–	0	1	–	–	0	1	0	1
Advanced	–	–	1	10	1	2	1	2	1	2	0	2	1	99	0	2
Intermediate	2	3	1	10	–	–	2	3	–	–	2	3	–	–	–	–
Novice	1	2	1	2	1	2	1	2	1	2	0	2	0	2	1	3
Dev. Devices	5	–	2	–	4	–	5	–	3	–	3	–	3	–	4	–
Test Devices	4	–	2	–	3	–	1	–	3	–	3	–	3	–	4	–
Females	1	–	1	–	1	–	1	–	1	–	1	–	1	–	1	–

D.6 Results: Stated Advantages and Disadvantages of TEASE

The following table shows the codes for the survey responses for advantages and disadvantages of TEASE, including number of occurrence of each code in the observational study described in Section 5.4.

Advantages	**Code**	**n**
Time savings thanks to algorithm	TIME	4
Suggestion gives a strong starting point for manual adaptations	SUGG	4
Large amount of information is displayed in a condensed form	INFO	5
Ability to take many criteria into account	CRIT	3
Less danger of forgetting to take a criterion into account	FORG	4
Less bias in the resulting assignment	BIAS	1
Disadvantages	**Code**	**n**
Can't find any or did not state any	NONE	3
Danger of overlooking the need for manual adaptation because of too much trust in algorithm	OVER	4
Qualitative criteria are not taken into account	QUAL	2
Project requirements are not taken into account	PROJ	1

List of Abbreviations

AI	Artificial Intelligence
FFT	Five-Factor Theory
GQM	Goal-Question-Metric Model
HR	Human Resources
MBTI	Myers-Briggs Type Indicator
TAM	Technology Acceptance Model
TEASE	**TE**am **A**llocator for **S**oftware **E**ngineering courses
TEMPO	**TE**am co**M**position in **P**roject-based **O**rganizations
TIPI	Ten-Item Personality Inventory
TTM	Technology Transfer Model
UI	User Interface
UML	Unified Modeling Language
UX	User Experience

List of Figures

List of Tables

Bibliography

[ADB16] L. Alperowitz, D. Dzvonyar, and B. Bruegge. “Metrics in Agile project courses.” In: *Proceedings of the 38th International Conference on Software Engineering Companion*. ACM Press, 2016, pp. 323–326.

[AJ04] S. T. Acuña and N. Juristo. “Assigning people to roles in software projects.” In: *Software: Practice and Experience* 34.7 (2004), pp. 675–696.

[AJDB17] L. Alperowitz, J. O. Johanssen, D. Dzvonyar, and B. Bruegge. “Modeling in Agile Project Courses.” In: *MODELS (Satellite Events)*. 2017, pp. 521–524.

[AP09] T. Ahtee and T. Poranen. “Risks in Students’ Software Projects.” In: *22nd Conference on Software Engineering Education and Training*. IEEE, 2009, pp. 154–157.

[ASS07] A. Aritzeta, S. Swailes, and B. Senior. “Belbin’s Team Role Model: Development, Validity and Applications for Team Building.” In: *Journal of Management Studies* 44.1 (2007), pp. 96–118.

[BA09] Belbin and B. Associates. *The Belbin guide to succeeding at work*. A&C Black, 2009.

[Bal06] H. H. Baligh. *Organization structures*. Springer, 2006.

[BBW08] A. Barreto, M. d. O. Barros, and C. M. L. Werner. “Staffing a software project: A constraint satisfaction and optimization-based approach.” In: *Computers & Operations Research* 35.10 (2008), pp. 3073–3089.

[BD09] B. Bruegge and A. H. Dutoit. *Object Oriented Software Engineering Using UML, Patterns, and Java (Third Edition)*. Prentice Hall International, 2009.

[Bee+08] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp. “Motivation in Software Engineering: A systematic literature review.” In: *Information and Software Technology* 50.9-10 (2008), pp. 860–878.

[Bel07] S. T. Bell. “Deep-level composition variables as predictors of team performance: A meta-analysis.” In: *Journal of Applied Psychology* 92.3 (2007), pp. 595–615.

[Bel12] R. M. Belbin. *Team roles at work*. Routledge, 2012.

[BG18] M. Bustamante and N. Gandhi. *Human resources in the age of automation.* 2018. URL: `https://tinyurl.com/y8hz48u6` (visited on January 2, 2018).

[BK13] M. Broy and M. Kuhrmann. *Projektorganisation und Management im Software Engineering.* Springer, 2013.

[BKW12] B. Bruegge, S. Krusche, and M. Wagner. "Teaching Tornado: from communication models to releases." In: *Proceedings of the 8th edition of the Educators' Symposium.* ACM, 2012, pp. 5–12.

[Bon10] D. A. Bonebright. "40 years of storming: a historical review of Tuckman's model of small group development." In: *Human Resource Development International* 13.1 (2010), pp. 111–120.

[Bos+11] I. Bosnić, I. Čavrak, M. Orlić, M. Žagar, and I. Crnković. "Student motivation in distributed software development projects." In: *Proceedings of the 2011 community building workshop on collaborative teaching of globally distributed software development.* ACM, 2011, pp. 31–35.

[BS06] J. M. Beaver and G. A. Schiavone. "The effects of development team skill on software product quality." In: *ACM SIGSOFT Software Engineering Notes* 31.3 (2006), pp. 1–5.

[BV04] S. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[BW84] V. R. Basili and D. M. Weiss. "A methodology for collecting valid software engineering data." In: *IEEE Transactions on software engineering* 6 (1984), pp. 728–738.

[CA10] L. Capretz and F. Ahmed. "Making Sense of Software Development and Personality Types." In: *IT Professional* 12.1 (2010), pp. 6–13.

[Cap03] L. F. Capretz. "Personality types in software engineering." In: *International Journal of Human-Computer Studies* 58.2 (2003), pp. 207–214.

[CCFC12] R.-C. Chen, S.-Y. Chen, J.-Y. Fan, and Y.-T. Chen. "Grouping partners for cooperative learning using genetic algorithm and social network analysis." In: *Procedia Engineering* 29 (2012), pp. 3888–3893.

[CCP90] T. D. Cook, D. T. Campbell, and L. Peracchio. "Quasi Experimentation." In: *Handbook of Industrial & Organizational Psychology.* Consulting Psychologists Press, 1990.

[Chu09] M. Y. Chuttur. "Overview of the technology acceptance model: Origins, developments and future directions." In: *Working Papers on Information Systems* 9.37 (2009), pp. 9–37.

[CL04] S.-J. Chen and L. Lin. “Modeling Team Member Characteristics for the Formation of a Multifunctional Team in Concurrent Engineering.” In: *IEEE Transactions on Engineering Management* 51.2 (2004), pp. 111–124.

[Cru+11] S. Cruz, F. da Silva, C. Monteiro, C. Santos, and M. dos Santos. “Personality in software engineering: preliminary findings from a systematic literature review.” In: *15th Annual Conference on Evaluation & Assessment in Software Engineering*. IET, 2011, pp. 1–10.

[CVL02] C. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2002.

[DAHB18] D. Dzvonyar, L. Alperowitz, D. Henze, and B. Bruegge. “Team Composition in Software Engineering Project Courses.” In: *Proceedings of the 2nd International Workshop on Software Engineering Education for Millennials*. 2018, pp. 16–23.

[Dan51] G. B. Dantzig. “Maximization of a linear function of variables subject to linear inequalities.” In: *Activity analysis of production and allocation* 13 (1951), pp. 339–347.

[Dav89] F. D. Davis. “Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology.” In: *MIS Quarterly* 13.3 (1989), pp. 319–340.

[DB18a] D. Dzvonyar and B. Bruegge. “Reaching Steady State in Software Engineering Project Courses.” In: *1st Workshop on Innovative Software Engineering Education*. 2018, pp. 8–11.

[DB18b] D. Dzvonyar and B. Bruegge. “Team Composition and Team Factors in Software Engineering: An Interview Study of Project-based Organizations.” In: *25th Asia-Pacific Software Engineering Conference*. Nara, 2018.

[DeF02] R. J. DeFillippi. “Organizational Models for Collaboration in the New Economy.” In: *Human Resource Planning* 25.4 (2002), pp. 7–18.

[DHAB18] D. Dzvonyar, D. Henze, L. Alperowitz, and B. Bruegge. “Algorithmically Supported Team Composition for Software Engineering Project Courses.” In: *IEEE Global Engineering Education Conference*. 2018, pp. 1753–1760.

[DKA14] D. Dzvonyar, S. Krusche, and L. Alperowitz. “Real Projects with Informal Models.” In: *Proceedings of the 10th edition of the Educators’ Symposium*. ACM, 2014, pp. 39–45.

[DLS06] E. Derby, D. Larsen, and K. Schwaber. *Agile retrospectives: Making good teams great*. Pragmatic Bookshelf, 2006.

[DM01] G. Dafoulas and L. Macaulay. “Facilitating group formation and role allocation in software engineering groups.” In: *Proceedings ACS/IEEE International Conference on Computer Systems and Applications*. IEEE, 2001, pp. 352–359.

[EW88] Y. M. Ermoliev and R.-B. Wets. *Numerical techniques for stochastic optimization*. Springer-Verlag, 1988.

[FA05] E. L. Fitzpatrick and R. G. Askin. “Forming effective worker teams with multi-functional skill requirements.” In: *Computers & Industrial Engineering* 48.3 (2005), pp. 593–608.

[FB14] N. Fenton and J. Bieman. *Software Metrics: A Rigorous and Practical Approach, Third Edition*. 3rd. Boca Raton, FL, USA: CRC Press, Inc., 2014.

[FHT01] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Vol. 1. 10. New York: Springer series in statistics, 2001.

[FPC01] S. Fincher, M. Petre, and M. Clark. *Computer science project work: principles and pragmatics*. Springer Science & Business Media, 2001.

[Fra+11] A. Franca, T. Gouveia, P. Santos, C. Santana, and F. da Silva. “Motivation in software engineering: a systematic review update.” In: *15th Annual Conference on Evaluation & Assessment in Software Engineering*. IET, 2011, pp. 154–163.

[FSM12] L. Fernández-Sanz and S. Misra. “Analysis of cultural and gender influences on teamwork performance for software requirements analysis in multinational environments.” In: *IET Software* 6.3 (2012), pp. 167–175.

[Gam95] E. Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[GFA09] T. Grossman, G. Fitzmaurice, and R. Attar. “A survey of software learnability: metrics, methodologies and guidelines.” In: *Proceedings of the 27th international conference on Human factors in computing systems*. New York, USA: ACM Press, 2009, pp. 649–658.

[GGLW06] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. “A Model for Technology Transfer in Practice.” In: *IEEE Software* 23.6 (2006), pp. 88–95.

[GL04] N. Gorla and Y. W. Lam. “Who should work with whom?” In: *Communications of the ACM* 47.6 (2004), pp. 79–82.

[Gra92] R. B. Grady. *Practical software metrics for project management and process improvement*. Prentice-Hall, Inc., 1992.

[HA08] P. L. Hunsaker and A. J. Alessandra. *The New Art of Managing People: Person-to-person Skills, Guidelines, and Techniques Every Manager Needs to Guide, Direct, and Motivate the Team*. Free Press, 2008.

[HA86] P. L. Hunsaker and A. J. Alessandra. *The Art of Managing People*. New York: Simon & Schuster, Inc., 1986.

[Har73] A. P. Hare. “Theories of group development and categories for interaction analysis.” In: *Small group behavior* 4.3 (1973), pp. 259–304.

[Har97] J. Harrison. “Enhancing software development project courses via industry participation.” In: *Proceedings of the 10th Conference on Software Engineering Education and Training*. IEEE, 1997, pp. 192–203.

[HC99] B. Hughes and M. Cotterrell. *Software project management*. 2nd ed. London: McGraw-Hill, 1999.

[Hen83] S. Henry. “A project oriented course on software engineering.” In: *Proceedings of the fourteenth SIGCSE technical symposium on Computer science education*. New York, USA: ACM, 1983, pp. 57–61.

[Her05] F. Herzberg. “Motivation-hygiene theory.” In: *Organizational behavior one: Essential theories of motivation and leadership* (2005), pp. 61–74.

[HG92] L Hirschhorn and T Gilmore. “The new boundaries of the "boundaryless" company.” In: *Harvard business review* 70.3 (1992), pp. 104–115.

[Hob00] M. Hobday. “The project-based organisation: an ideal form for managing complex products and systems?” In: *Research policy* 29.7-8 (2000), pp. 871–893.

[Hoc95] D. W. Hock. “The chaordic organization: Out of control and into order.” In: *World Business Academy Perspectives* 9.1 (1995), pp. 5–18.

[JBR99] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[Jon10] G. Jones. *Organizational Theory, Design and Change*. Sixth Edit. Upper Saddle River, NJ, USA: Pearson Education, 2010.

[KABW14] S. Krusche, L. Alperowitz, B. Bruegge, and M. O. Wagner. “Rugby: an agile process model based on continuous delivery.” In: *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*. ACM, 2014, pp. 42–50.

[KDXB18] S. Krusche, D. Dzvonyar, H. Xu, and B. Bruegge. “Software Theater - Teaching Demo Oriented Prototyping.” In: *ACM Transactions on Computing Education* 18.2 (July 2018), 10:1–10:30.

[KM16] M. Kuhrmann and J. Münch. “When teams go crazy.” In: *Proceedings of the 38th International Conference on Software Engineering Companion*. New York, USA: ACM Press, 2016, pp. 412–421.

[KS93] J. R. Katzenbach and D. K. Smith. “The discipline of teams.” In: *Harvard Business Review* 71.2 (1993), pp. 111–120.

[KW97] S. L. Kichuk and W. H. Wiesner. “The big five personality factors and team performance: implications for selecting successful product design teams.” In: *Journal of Engineering and Technology Management* 14.3-4 (1997), pp. 195–221.

[KYR06] H.-R. Kang, H.-D. Yang, and C. Rowley. “Factors in team effectiveness: Cognitive and demographic similarities of software development team members.” In: *Human Relations* 59.12 (2006), pp. 1681–1710.

[Lal14] F. Laloux. *Reinventing Organizations: Ein Leitfaden zur Gestaltung sinnstiftender Formen der Zusammenarbeit*. Munich, 2014.

[LB02] R. Lingard and E. Berry. “Teaching teamwork skills in software engineering based on an understanding of factors affecting group performance.” In: *32nd Annual Frontiers in Education*. IEEE, 2002, pp. 1–6.

[LLLL07] T.-P. Liang, C.-C. Liu, T.-M. Lin, and B. Lin. “Effect of team diversity on software project performance.” In: *Industrial Management & Data Systems* 107.5 (2007), pp. 636–653.

[LLOR10] R. A. Layton, M. L. Loughry, M. W. Ohland, and G. D. Ricco. “Design and Validation of a Web-Based System for Assigning Members to Teams Using Instructor-Specified Criteria.” In: *Advances in Engineering Education* 2.1 (2010), pp. 1–28.

[LPM09] S. Licorish, A. Philpott, and S. G. MacDonell. “Supporting agile team composition: A prototype tool for identifying personality (In)compatibilities.” In: *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*. IEEE, 2009, pp. 66–73.

[LS17] F. S. Lobato and V. Steffen. “Multi-Objective Optimization Problem.” In: *Multi-Objective Optimization Problems*. Springer, 2017, pp. 9–23.

[Mad07] R. J. Madachy. *Software process dynamics*. John Wiley & Sons, 2007.

[Map88] M. F. Maples. “Group development: Extending tuckman’s theory.” In: *The Journal for Specialists in Group Work* 13.1 (1988), pp. 17–23.

[Mar01] C. Margerison. “Team competencies.” In: *Team Performance Management: An International Journal* 7.7/8 (2001), pp. 117–122.

[McG60] D. McGregor. “Theory X and theory Y.” In: *Organization theory* 358 (1960), pp. 374–378.

[MCJ99] R. R. McCrae and P. T. Costa Jr. “A five-factor theory of personality.” In: *Handbook of personality: Theory and research* 2.1999 (1999), pp. 139–153.

[MHS13] M. B. Miles, A. M. Huberman, and J. Saldana. *Qualitative data analysis*. Sage, 2013.

[Mie12] K. Miettinen. *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012.

[MM06] K. Miettinen and M. M. Mäkelä. "Synchronous approach in interactive multiobjective optimization." In: *European Journal of Operational Research* 170.3 (2006), pp. 909–922.

[MM10] I. B. Myers and P. B. Myers. *Gifts differing: Understanding personality type*. Nicholas Brealey, 2010.

[MM90] C. J. Margerison and D. McCann. *Team management: Practical new approaches*. Mercury Books, 1990.

[MRF15] G. Matturro, F. Raschetti, and C. Fontán. "Soft skills in software development teams: A survey of the points of view of team leaders and team members." In: *Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE Press. 2015, pp. 101–104.

[MSG93] B. B. Morgan Jr, E. Salas, and A. S. Glickman. "An analysis of team evolution and maturation." In: *The Journal of General Psychology* 120.3 (1993), pp. 277–291.

[MT04] J. E. McGrath and F. Tschan. *Temporal matters in social psychology: Examining the role of time in the lives of groups and individuals*. American Psychological Association, 2004.

[NCY02] J. Nielsen, T. Clemmensen, and C. Yssing. "Getting access to what goes on in people's heads?" In: *Proceedings of the second Nordic conference on Human-computer interaction*. New York, USA: ACM Press, 2002, pp. 101–110.

[Nie94] J. Nielsen. *Usability engineering*. Morgan Kaufmann, 1994.

[NWC99] G. A. Neuman, S. H. Wagner, and N. D. Christiansen. "The relationship between work-team personality composition and the job performance of teams." In: *Group & Organization Management* 24.1 (1999), pp. 28–45.

[Ojh05] A. K. Ojha. "Impact of team demography on knowledge sharing in software project teams." In: *South Asian Journal of Management* 12.3 (2005), pp. 67–78.

[OR12] P. O'Leary and I. Richardson. "Process reference model construction: implementing an evolutionary multi-method research approach." In: *IET Software* 6.5 (2012), pp. 423–430.

[Pee06] M. A. G. Peeters. “The Big Five Personality Traits and Individual Satisfaction With the Team.” In: *Small Group Research* 37.2 (2006), pp. 187–211.

[PFP11] G. Purna Sudhakar, A. Farooq, and S. Patnaik. “Soft factors affecting the performance of software development teams.” In: *Team Performance Management: An International Journal* 17.3/4 (2011), pp. 187–205.

[PRW13] A. Picot, R. Reichwald, and R. T. Wigand. *Die grenzenlose Unternehmung: Information, Organisation und Management. Lehrbuch zur Unternehmensführung im Informationszeitalter*. Springer-Verlag, 2013.

[PSCB00] C. R. Paris, E. Salas, and J. A. Cannon-Bowers. “Teamwork in multi-person systems: a review and analysis.” In: *Ergonomics* 43.8 (2000), pp. 1052–1075.

[PSE04] D. E. Perry, S. E. Sim, and S. M. Easterbrook. “Case studies for software engineers.” In: *Proceedings. 26th International Conference on Software Engineering*. IEEE. 2004, pp. 736–738.

[Ree99] J. Reel. “Critical success factors in software projects.” In: *IEEE Software* 16.3 (1999), pp. 18–23.

[RH09a] P. Runeson and M. Höst. “Guidelines for conducting and reporting case study research in software engineering.” In: *Empirical Software Engineering* 14.2 (2009), pp. 131–164.

[RH09b] P. Runeson and M. Höst. “Guidelines for conducting and reporting case study research in software engineering.” In: *Empirical Software Engineering* 14.2 (2009), pp. 131–164.

[RHRR12] P. Runeson, M. Höst, A. Rainer, and B. Regnell. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons, 2012.

[RIRJSV10] J. G. Rivera-Ibarra, J. Rodríguez-Jacobo, and M. A. Serrano-Vargas. “Competency Framework for Software Engineers.” In: *23rd IEEE Conference on Software Engineering Education and Training*. IEEE, 2010, pp. 33–40.

[RJ15] S. P. Robbins and T. A. Judge. *Organizational behavior*. 16th ed. Boston, MA: Pearson, 2015.

[Rob07] B. J. Robertson. “Organization at the leading edge: Introducing Holacracy^{TM}.” In: *Integral Leadership Review* 7.3 (2007), pp. 1–13.

[RS14] T. Rhein and H. Stüber. *Beschäftigungsdauer im Zeitvergleich: Bei Jüngeren ist die Stabilität der Beschäftigung gesunken*. Tech. rep. Institut für Arbeitsmarkt- und Berufsforschung, 2014. URL: `https://tinyurl.com/ybls9kpf` (visited on April 28, 2018).

[RVGFLR12] E. Romero, P. Villar, J. A. Gómez-Fraguela, and L. López-Romero. “Measuring personality traits with ultra-short scales: A study of the Ten Item Personality Inventory (TIPI) in a Spanish sample.” In: *Personality and Individual Differences* 53.3 (2012), pp. 289–293.

[SBCR02] R. van Solingen, V. Basili, G. Caldiera, and H. D. Rombach. “Goal Question Metric (GQM) Approach.” In: *Encyclopedia of Software Engineering*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2002.

[SG10] D. Strnad and N. Guid. “A fuzzy-genetic decision support system for project team formation.” In: *Applied Soft Computing* 10.4 (2010), pp. 1178–1187.

[SG74] R. C. Sarri and M. J. Galinsky. “A conceptual framework for group development.” In: *Individual change through small groups* (1974), pp. 71–88.

[SLPK15] K. Schneider, O. Liskin, H. Paulsen, and S. Kauffeld. “Media, Mood, and Meetings.” In: *ACM Transactions on Computing Education* 15.4 (2015), pp. 1–33.

[SMG11] N. Salleh, E. Mendes, and J. Grundy. “The effects of openness to experience on pair programming in a higher education context.” In: *24th Conference on Software Engineering Education and Training*. IEEE, 2011, pp. 149–158.

[Smi01] G. Smith. “Group development: A review of the literature and a commentary on future research directions.” In: *Group Facilitation* 3.Spring (2001), pp. 14–45.

[Ste06] G. L. Stewart. “A meta-analytic review of relationships between team design features and team performance.” In: *Journal of Management* 32.1 (2006), pp. 29–55.

[Ste16] StepStone. *Jobs nach Maß: Was Fachkräfte wollen*. Tech. rep. 2016. URL: `https://tinyurl.com/ydz25yo2` (visited on February 4, 2019).

[THCG04] T.-L. B. Tseng, C.-C. Huang, H.-W. Chu, and R. R. Gung. “Novel approach to multi-functional project team formation.” In: *International Journal of Project Management* 22.2 (2004), pp. 147–159.

[TJ77] B. W. Tuckman and M. A. C. Jensen. “Stages of small-group development revisited.” In: *Group & Organization Studies* 2.4 (1977), pp. 419–427.

[VBS94] M. W. Van Someren, Y. F. Barnard, and J. A. C. Sandberg. *The think aloud method: a practical approach to modelling cognitive processes*. Academic Press, 1994.

[VP95] A. H. de Ven and M. S. Poole. “Explaining development and change in organizations.” In: *Academy of management review* 20.3 (1995), pp. 510–540.

[Woh+12] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[WWB07] T. van de Water, H. van de Water, and C. Bukman. "A balanced team generating model." In: *European Journal of Operational Research* 180.2 (2007), pp. 885–906.

[YC11] I.-T. Yang and J.-S. Chou. "Multiobjective optimization for manpower assignment in consulting engineering firms." In: *Applied Soft Computing* 11.1 (2011), pp. 1183–1190.

[Yin17] R. K. Yin. *Case study research and applications: Design and methods*. Sage publications, 2017.

[YKM08] H.-D. Yang, H.-R. Kang, and R. M. Mason. "An exploratory study on meta skills in software development teams: antecedent cooperation skills and personality for shared mental models." In: *European Journal of Information Systems* 17.1 (2008), pp. 47–61.

[Yuk13] G. Yukl. *Leadership in organizations*. 8th ed. Harlow: Pearson Education Limited, 2013.

[Bur18] Bureau of Labor Statistics. *Employee Tenure in 2018*. Tech. rep. U.S. Department of Labor, 2018. URL: `https://tinyurl.com/ycwoxnhf` (visited on March 17, 2019).

[Hum17] Human Resources Professionals Association. *A New Age of Opportunities: What does Artificial Intelligence mean for HR Professionals?* Tech. rep. 2017. URL: `https://tinyurl.com/y8e4kq2n` (visited on March 17, 2019).

[IBM16] IBM Institute for Business Value. *Extending expertise: How cognitive computing is transforming HR and the employee experience*. Tech. rep. 2016. URL: `https://tinyurl.com/y7ono4a5` (visited on March 17, 2019).