Gottfried Wilhelm
Leibniz Universität Hannover
Faculty for Elektrotechnik und Informatik
Institute for Practical Informatics
Software Engineering Group

Deriving a Quality Model for Collaboration in Software Projects Based on an Interview Study and Literature

Master's Thesis

in the Major of Computer Science

by

Michael Mircea

Examiner: Prof. Dr. Kurt Schneider
Secondary Examiner: Dr. Jil Klünder
Tutor: Alexander Specht

Hannover, 01.08.2024

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 01.08.2024

Michael Mircea

Zusammenfassung

Zusammenarbeit in Softwareteams ist ein komplexer, vielseitiger und ausschlaggebender Faktor für den Erfolg von Projekten und die Zufriedenheit deren Entwickler. Qualitätsmodelle werden zahlreich im Bereich der Softwareentwicklung genutzt, um komplexe Attribute zu verbessern, allerdings hauptsächlich um die Qualität der entwickelten Software messbar zu machen. Diese Arbeit verfolgt den Ansatz die Funktionalität von Qualitätsmodellen zu nutzen, um den komplexen Begriff der Zusammenarbeit in Softwareteams in einzelne Qualitätsaspekte zu unterteilen und diese messbar und verbesserbar zu machen. Dabei liegt ein besonderer Fokus darauf, dass das Modell universell einsetzbar ist, unabhängig von der genauen Art der Zusammenarbeit. Das hier entwickelte Qualitätsmodell basiert auf der Analyse einer umfangreichen Interviewstudie und einer ausführlichen Literaturrecherche. Das entwickelte Modell wurde in einer weiteren Interviewstudie hinsichtlich ihrer Vollständigkeit validiert. Basierend auf den Interviews wurden Anpassungen am Model vorgenommen. Anschließend wurden quantitative Daten in einer Onlineumfrage erhoben, um die Wichtigkeit und Korrektheit der einzelnen Qualitätsaspekte zu ermitteln. Abschließend wurde die Validität diskutiert. Das Qualitätsmodell bietet somit eine fundierte Grundlage sowohl zur direkten Anwendung in Softwareteams als auch für eine fortführende Anpassung und Weiterentwicklung.

Abstract

Collaboration in software teams is a complex, broad and decisive factor for project success and developer satisfaction. Quality models are widely used in the domain of software engineering to make complex attributes measurable, but mainly in order to improve the quality of the developed software itself. This thesis follows an approach of trying to use the functionality of quality models in order to subdivide the complex concept of collaboration in software teams into distinct quality aspects and make them both measurable and improvable. A special emphasis was put on making the model as universally applicable as possible, regardless of the exact mode of collaboration. The developed quality model is based on an analysis of an extensive interview study, as well as a detailed literature review. The developed model was validated for completeness in a further interview study. Based on the findings, the model was adjusted. Subsequently, quantitative data was gathered in an online survey in order to determine the correct placement and importance of the single quality aspects. Lastly, all findings were discussed in regards to the model's validity. Thus the quality model offers a well-founded basis, both for the direct application in software teams as well as for further adjustments or extensions.

Contents

Chapter 1

Introduction

Researchers have been trying to analyze the aspect of collaboration in software development for a long time. In a project from 1999, the results of which were published in a paper from 2002 by Augustin et al. [1], a first attempt was made to determine the reason why internet developed community projects of the time (such as Linux, Apache and Samba) were developed faster and with higher quality than comparable commercially available alternatives. From those successful projects they derived a set of practices they coined *Collaborative Software Development* (CSD). These practices mostly opposed the mainstream organizational culture of the time and are congruent with the simultaneously emerging agile principles, as they are known from the *Agile Manifesto* [3] today. They also showed the first signs of special needs and attentions in regards to collaboration in software development, which diverged from the type of collaboration that was viewed as the norm.

Ever since then, efforts have been made to better understand the aspect of collaboration in software teams. However, at the same time collaboration in the domain of software development grew more and more complex, making it ever harder to define.

1.1 Problem Statement

The quality of collaboration is highly correlated with the quality of the developed software and developer satisfaction [12]. However, collaboration in software teams is a more broad and multi-faceted concept today than ever, because of the increases in temporal, geographical and sociocultural distances [15] within software teams. In addition, the term *collaboration* is widely used in research, however it is rarely clearly and consistently defined and its meaning is often assumed to be obvious, both in literature [2] and in practice [24]. It is, for example, often used as a synonym for *teamwork*,

or just generally not considered separately from other social factors [24]. This shows a clear need for a widely applicable, but also detailed approach to defining what constitutes collaboration in software teams.

The research questions of this work, as guided by the described problems, are as follows:

RQ1: What constitutes high quality collaboration in software teams?

RQ2: How can the quality of collaboration in software teams be measured and improved?

1.2 Solution Approach

Quality models (cf. section 2.1) have established themselves as valuable tools in the domain of software engineering [26]. They are especially useful for breaking down a complex concept into simpler subdimensions, which are measurable and the improvement of which contributes to the improvement of the complex aspect. They are used numerously, however mostly to assess the quality of the developed software itself. This work follows an approach of utilizing the functionality of a quality model in order to define the aspect of collaboration, specifically in the domain of software development, and make its quality more objective and quantifiable. A special focus was set on accommodating all types of collaboration in software teams, regardless of temporal, geographical or sociocultural distances, while overfitting the model to none of them.

1.3 Results

This work derived a quality model for collaboration in software teams through an extensive literature review, supplemented by industry expert opinions and validation. The aspect of *Collaboration* was eventually broken down into six subdimensions with a total of 17 quality aspects. A first draft of the model was validated for completeness in expert interviews. These interviews offered additional insights to the literature, which led to slight alterations to the model. The adjusted model was then validated in an online survey which was exclusively administered to software developers with experience of working in teams. A total of 58 subjects completed the survey, which led to quantifiable data about the importance of the individual quality aspects in regards to collaboration and further modifications. In addition to the scope of regular quality models, this work also suggests metrics for measuring each of the quality aspects, which are mostly established in

scientific literature and supplemented with the data gathered in the interview and survey. Based on holistic discussion of all evidence, the model offers a well-founded basis for direct application in software teams. However, there still remains plenty of potential for future work in regards to the model, which will be discussed at the end.

1.4 Structure

This work is structured as follows: The following chapter 2 will go over the fundamentals pertaining to this thesis, giving explanations of used terms and concepts. It also gives an overview of related work that was most influential for the creation of the model. Chapter 3 uses the gathered information and presents the argumentation for the synthesis of the quality model. Subsequently, the validation of the model is presented. First, the interview study, its findings and the resulting changes to the model are presented in chapter 4. Subsequently the quantitative data gathered by the online survey is presented in chapter 5. All the previously mentioned findings and their final implications for the model are holistically discussed in chapter 6. Afterwards, chapter 7 supplements the validated model with metric suggestions for the individual quality aspects. Finally, the last chapter 8 gives a closing review and an outlook on possible future work that could be conducted with the findings of this thesis.

Chapter 2

Fundamentals

This chapter offers a familiarization with the concepts commonly used and referenced within this work. It will offer a brief introduction into the fundamentals necessary to understand the later chapters, as well as an overview of related work that was influential to the developed model. In order to keep this chapter concise, the related works will only be presented partially and superficially within this chapter, while more specific findings relevant to the model will be discussed in detail in chapters 3 and 7.

2.1 Quality Models and GQM

Quality models are an option to achieve a measurement mechanism for evaluating the quality of an object of observation in software engineering. They have been tested numerously in studies to evaluate quality within the domain of software engineering [26][23]. Most of these models seek to analyze and improve the quality of the software product itself, including the most well known quality model which is documented in the *International Organization for Standardization's* ISO 25010 [13] standard. A partial (for the purposes of visibility) visualization of the model is presented in figure 2.1. While the quality model developed in this thesis will not concern itself with the quality of the software product (though correlations between quality of collaboration and quality of a software product have been shown [12]), ISO 25010 lends itself well for the explanation of quality models and how they are used. Software quality is a very complex, broad, subjective and hard to measure concept. There are many factors that need to be considered when talking about the quality of a software. ISO 25010 deconstructs the concept into its characteristics (referred to as subdimensions within this work), for example *Functional Suitability*. These subdimensions all contribute to the quality of the software as a whole, however they are already less complex. In the next step, the model deconstructs the subdimensions into further sub-characteristics (referred to

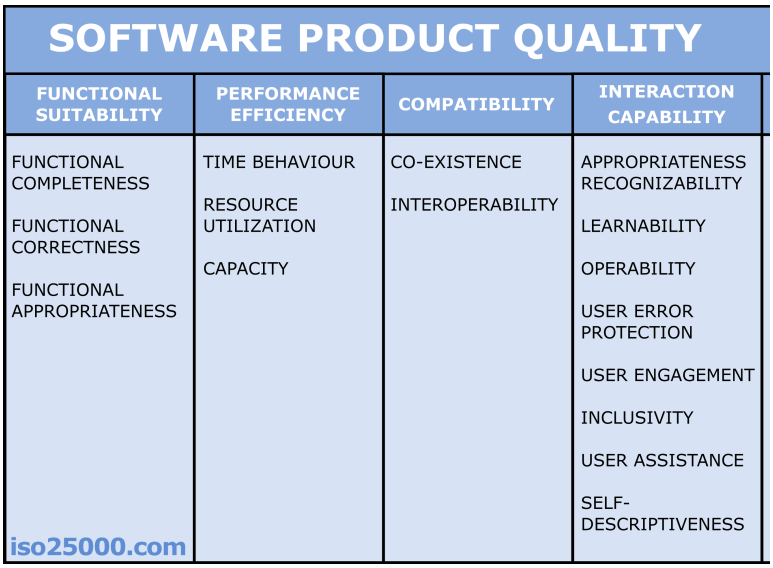


Figure 2.1: Excerpt of ISO 25010, limited to four (out of nine) subdimensions for better visibility. Modified from [13].

as quality aspects within this work). In the case of *Functional Suitability*, it is further deconstructed into *Functional Completeness*, *Functional Correctness* and *Functional Appropriateness*. This process can be repeated almost indefinitely to deconstruct the subdimensions into further and further subdimensions. However, the goal is to arrive at quality aspects that are simple and specific enough to be measured by specified metrics. The analysis, measurement and improvement of these metrics then contribute to the improvement of the higher dimensions and finally the software product quality itself. As mentioned before, software quality can be highly subjective and case specific. For example, every project has its own definition of what constitutes *Functional Appropriateness*. Thus, determining the metrics by which to measure the quality aspects is part of using a quality model in practice.

An approach for deriving the appropriate metrics for the specific case is the *Goal Question Metric Approach* [7] (GQM). The authors describe the approach as follows:

"The Goal Question Metric (GQM) approach is based upon the assumption that for an organization to measure in a purposeful way it must first specify the goals for itself and its projects, then it must trace those goals to the data that are intended to define those goals operationally, and finally provide a framework for interpreting the data with respect to the stated goals."

This process will be explained with the example of an online learning tool for students: First a goal is defined for an object. When using a quality model, the object in question will be a quality aspect of the model, such as *Capacity* (part of *Performance Efficiency*) in ISO 25010. In our hypothetical scenario the goal could be defined as follows:

> **Goal:** *"The capacity of the online tool needs to be able to accommodate our current student numbers during peak hours".*

Next, a set of questions is defined to characterize the way a specified goal can be achieved. Example:

> **Question 1:** *"Are there any server outages or connection aborts due to overload?"*

Lastly, a set of metrics can be derived to quantitatively answer every question. Example:

> **Metric 1:** *Average number of connection aborts in a 24 hour period.*
> **Metric 2:** *Average server downtime (in seconds) in a 24 hour period.*

Thus, a quality model can be used to analyze a complex aspect, deconstruct it into measurable quality aspects and determine metrics to measure and improve them for the specific goals of a project. While GQM is a common practice in the application of a quality model, this thesis will further supplement its quality model with generic metric suggestions for the measurement of its quality aspects in chapter 7.

2.2 Collaboration versus Teamwork

Part of the motivation for this thesis was the often unclear and inconsistent definition of the term *collaboration*, even in scientific literature. In their 2017 paper *"Collaboration in agile software development: Concept and dimensions"* [2], Batra et al. note the following:

> *"Indeed, in searching the literature, we found numerous studies that mention the term. However, these studies often assume what collaboration means, do not provide clear and consistent definitions, and rarely elaborate on the dimensions that constitute collaboration."*

Similar experiences were made during the literature review for this paper. Furthermore, in an interview study [24], practicing software developers mostly failed to define the term collaboration by itself, instead equating it with other socio-technical factors like communication or motivation.

They often described these factors as being "basically the same" or "going hand in hand", thereby showing the need for clarification, but also the strong interdependence between these factors, which is something that will be a recurring, important fact considered within this work. Therefore, the first step in creating the quality model was to create a clear distinction between *collaboration* and other, related terms.

The distinction between collaboration referring to developer teams and the frequently used related terms *customer collaboration* and *teamwork* were especially challenging in literature. In a highly influential work from 2001 [12] Hoegl and Gemuenden try to define factors that constitute high quality teamwork. However, the authors explicitly and exclusively use the term *Teamwork Quality* (TWQ) as a measure of collaboration in teams. In this case, the terms are used interchangeably. Other authors, however, ascribe further qualities to teamwork than factors relevant to collaboration. In a 2017 paper [28], Weimar et al. extend and validate the aforementioned TWQ model. Among other things, they extend the model with a factor called *Coordination of Expertise*. This is a management factor concerning itself with efficient resource management, without being directly related to collaboration. Similarly, in a paper analyzing teamwork effectiveness in agile software development [27], Strode et al. developed a Teamwork Effectiveness Model (ATEM). They included the factor *Redundancy*, derived from the factor *Backup Behavior* (taken from the teamwork model [22] by Salas et al.). *Redundancy* and *Backup Behavior* refer to the *"ability to shift workload among members to achieve balance during high periods of workload or pressure"* [22]. For example, if an important developer is overloaded, or temporarily unfit for work, somebody needs to be able to take over his responsibilities. This might be good teamwork, but no collaboration is involved in the process. Conclusively, it is also concerned with efficient resource management and almost counterproductive to collaboration, as it seeks to redistribute a workload to different individuals.

The *Merriam Webster*[1] dictionary definitions of the two terms further supplement the distinction that will eventually be made for the scope of this work:

- Teamwork: *work done by a group acting together so that each member does a part that contributes to the efficiency of the whole*
- to collaborate: *to work jointly with others or together especially in an intellectual endeavor*

To summarize, the concept of teamwork concerns itself with all work done by individuals as part of a team, which contributes to the efficiency of

[1] https://www.merriam-webster.com (accessed 24.06.2024)

the team as a whole. As such, teamwork contains collaboration between team members as a subset of qualities. However, it also consists of work done by individuals in solidarity, but as part of a team, such as filling vacancies to evenly distribute the workload. Since teamwork mostly concerns itself with efficiency and results of the team, concepts such as efficient resource management or coordination of expertise are relevant factors for its quality. However, these factors are not relevant to the quality of the collaboration itself. In contrast to the definition of teamwork, the definition for *collaboration* within the scope of this thesis will be centered around interactions and is as follows:

"**Collaboration** is the **process** of **two or more** individuals **interacting**, directly or indirectly, for the purpose of **achieving a shared task or goal**."

Collaboration can exist cross-functionally (such as with *customer collaboration*) within teams or between teams. As this work seeks to improve the collaboration of software teams, the main focus for the quality model will be on intra-team collaboration. Despite the high quality and influence of research in regards to teamwork quality, this work will not consider factors that do not directly impact the quality of collaboration as defined above.

2.3 State of Collaboration in Software Teams

Due to its digital nature, software development has always been a type of work well suited for distributed teams. Going back to the late 90s, open source online projects have already proven themselves capable of producing high quality software [1]. Nowadays, working across distances has become a common practice in software development [15], both through globally distributed teams, remote work, as well as the rising popularity of home office options in employment, especially following the COVID-19 pandemic [10]. Indeed, in searching the literature, many papers analyzing collaboration did so for a special mode of collaboration, such as open source development [1] or globally distributed organizational teams [15].

When designing a model for collaboration in software teams, one has to take into account all possible modes of collaboration that are commonplace and consider their impact on the teams' collaboration. Lanubile et al. [15] analyzed the threats of globally distributed development on certain team related factors. They summarized their findings in figure 2.2. The table shows that there are three dimensions to distance in software teams: Temporal distance, geographical distance and sociocultural distance. Especially the factors *communication* and *coordination* are highly relevant to collaboration. For example, a high temporal distance reduces opportunities

	Temporal Distance	Geographical Distance	Sociocultural Distance
Communication	- Reduced opportunities for synchronous communication	- Face-to-face meetings difficult	- Cultural misunderstandings
Coordination	- Typically increased coordination costs	- Reduced informal contact can lead to lack of critical task awareness	- Inconsistent work practices can impinge on effective coordination - Reduced cooperation arising from misunderstandings
Control	- Management of project artifacts may be subject to delays	- Difficult to convey vision and strategy - Perceived threat from training low-cost "rivals"	- Different perceptions of authority can undermine morale - Managers must adapt to local regulations

Figure 2.2: Impact of dimensions of distance on team related factors [15].

for synchronous communication, while geographical distance generally eliminates the possibility of face-to-face meetings. In an interview study [24], practicing software developers overwhelmingly noted this as harmful for the quality of collaboration and other social factors. This sentiment was echoed by the interview study conducted as part of this work (cf. chapter 4).

The presented issues add further complexity in the pursuit of designing a quality model that is universally applicable to collaboration in software teams. Conclusively, it is most important to determine the factors that all modes of collaboration have in common and are generally improvable. For example, evidence might suggest that face-to-face communication is more conducive to high quality collaboration than online meetings. However, a team being geographically distanced, or the high employee demand for home office work, are not facts that can simply be changed. Similarly, sociocultural distance between team members may present difficulties for collaboration, but cannot be eliminated in cases when teams are globally distributed. Consequently, including aspects like communication richness, or prevalence of face-to-face meetings as a quality aspect in the model would reduce its usefulness for many teams. Instead, improvable commonalities between all these modes of communication are sought out. For example, the quality of *Information Exchange* and *Shared Understanding* (cf. chapter 3) within all of those teams are aspects which can be analyzed, measured and improved. Therefore they are much better suited as aspects within a universally applicable quality model.

Lastly, the existence of both agile and traditional teams needs to be acknowledged. Though these two types of teamwork differ in terms of methods and emphasis on collaboration, Lindsjorn et al. [16] found that collaboration quality (measured through TWQ) and its effects are not greater in agile teams than in traditional ones. This is discussed in more detail in section 2.4.2.

2.4 Related Work

Many works had an influence on this thesis, however a select few offered large amounts of contributions to the eventual model. This section serves the purpose of briefly outlining those works, crediting their contributions and highlighting the key differences, demarcating this work from the existing literature.

2.4.1 Interview Study about Social Factors by Sasse

In his 2023 master's thesis, Sasse conducted an interview study about social factors in modern software projects [24], prompting the initial motivation for the research questions that eventually became the center of this work. The author interviewed 20 practicing German software developers, which were mostly biased towards being project leads. Since the conclusions presented in the paper were mainly about social factors in general and not merely about collaboration, all interview transcripts were read in full to gain first insights about industry expert opinion on collaboration. Consequently, most of the following quotes are taken directly from the raw interview transcripts and will not be found in the paper. Furthermore, all quotes have been translated from German.

Many participants made statements within the interviews that denoted the importance of collaboration. One participant compared the importance of the quality of collaboration with the quality of the individual:

"The quality of the individual and the collaboration of the others is almost on the same level."

Another participant highlighted that problems with collaboration lead to blockages of the entire project:

"I always work in teams and when the collaboration doesn't fit, then nothing works. I mean, then just nothing progresses and thus collaboration is the most important[...]."

Furthermore, the author asked the participants to rank social aspects in terms of importance. The results are shown in table 2.1. As can be seen, collaboration is ranked the second most important social aspect by the participants, with 75% of participants regarding it as important.

Social Aspect	**Amount**
Communication	19
Collaboration	15
Trust	10
Team leading	9
Error handling	9
Motivation	6
Relationship with customer	6
Knowledge transfer	4
Conflict handling	2
Cultural factors	0

Table 2.1: Importance of social factors in modern software projects. Translated and modified from [24].

Another very important finding from the interviews, which was also explicitly noted within the work, was the strong interdependence of social factors, including collaboration. Participants frequently stated that factors are dependent on one another, or even equating two of them. Two participants noted the requirement of trust for collaboration:

"Well, trust and collaboration are almost the same point here. You need a certain base trust to conduct a collaboration."

"Collaboration and trust are in very strong relation. Well, they are almost inseparably linked, collaborating in a trustful relation is one aspect for now."

Another participant noted how good collaboration automatically leads to the improvement of other factors:

"I've long thought about choosing knowledge transfer [in the ranking of social factors]*, but I reckon it comes automatically through collaboration."*

While some participants talked about how good collaboration can and should be encouraged, others took good collaboration for granted. One participant noted, that collaboration just "has to work" in a professional relationship, even if the people do not like working with each other. Another participant also took high quality collaboration for granted when other factors are given, again demonstrating interdependence, but also showing a lack of nuance when it comes to all the factors that constitute collaboration:

"Trust, motivation. If I have those, then the collaboration is good anyway, right?"

This was just an excerpt of insights about importance of collaboration and the impact of other social factors on collaboration gained from the interview transcripts. These insights were highly influential in the first shaping of the quality model and will be referenced again in chapter 3, since they offer partial justification for the presence of certain aspects within the quality model.

2.4.2 Teamwork Quality (TWQ) and related studies

Hoegl and Gemuenden's work regarding teamwork quality factors was the basis for much research into the topic. In their 2001 paper *"Teamwork Quality and the Success of Innovative Projects"* [12], the authors seek to answer the following questions: *"What is teamwork and how can it be measured? Why and how is teamwork related to the success of innovative projects? How strong is the relationship between teamwork and various measures of project success such as performance of team member satisfaction?"* As discussed in 2.2, the authors use the terms *teamwork* and *collaboration* interchangeably, describing TWQ as being a comprehensive concept of the collaboration in teams. Thus their findings are extremely relevant to this work. Even more relevance comes from their validation: They tested the relationship between TWQ and project success, as well as developer satisfaction using data from 575 team members of 145 German software teams. As such, their work presents strong evidence for assumptions about collaboration in software teams.

The authors determined six facets with which to describe the quality of teamwork: Communication, Coordination, Balance of Member Contributions, Mutual Support, Effort and Cohesion, shown with brief descriptions in figure 2.3. They reported the following findings regarding TWQ: Firstly, the quality of collaboration in teams can be captured through the six facets of TWQ. Secondly, they found a relationship between TWQ and the success of innovative projects, measured through effectiveness and efficiency of team performance, as well as the personal success of team members (measured through satisfaction and learning). These two findings already cement the six TWQ facets as mandatory considerations for the quality model of this work. Lastly, they found that the magnitude of relationship between TWQ and team performance varied with the perspective of the rater (for instance team member vs team leader). That being said, their work also contains certain limitations and shortcomings:

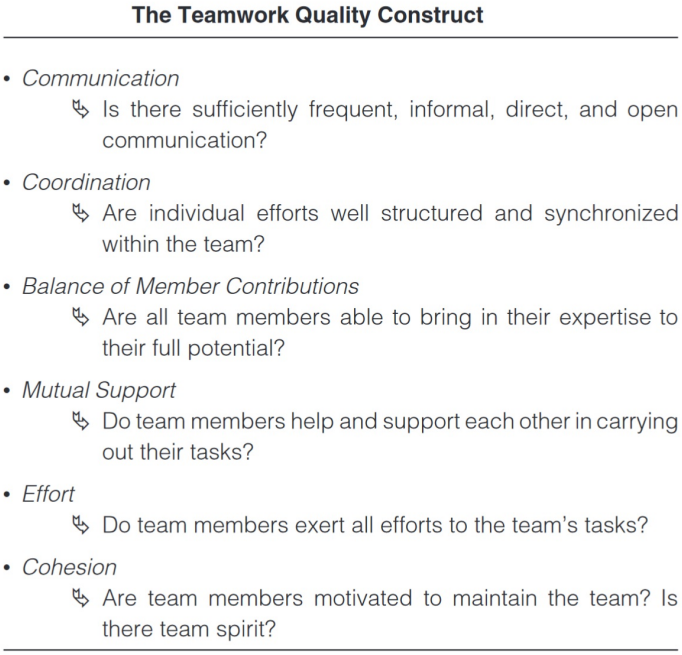


Figure 2.3: Hoegl and Gemuenden's six facets for Teamwork Quality [12].

First, all of the facets were measured using standardized questionnaires (five-point answer scale), with three to ten items per facet, which is a comparatively subjective metric. The problem with this arises when looking at their independent variable measurements: Measures of team performance and personal success of team members was also measured using questionnaire scales, using the same raters (with the exception of customer rating for project effectiveness). Thus, the authors themselves note, that they cannot establish a causality between TWQ and the outcome factors. Therefore, the relation could have simply been found due to the perception of well-functioning teamwork and the perception of a good result being highly interlinked in the eyes of the raters (which were members, leaders and managers of the team themselves). The metrics suggested for the quality model in chapter 7 demonstrate, that there are more objective ways to measure some of the collaboration facets in software teams (though for the more social-sided aspects questionnaire scales cannot be avoided).

Secondly, while the facets were validated in software teams, they are highly generic and do not take into account special needs of collaboration specifically within the domain of software engineering, many of which only came to light after the time of the study. In contrast, the quality model developed within this work is highly specified to software teams, with some quality aspects even being entirely exclusive to software engineering.

Lastly, it is assumed that the model was validated in exclusively traditional teams (in contrast to agile ones). This assumption is made because of the fact that the study was conducted shortly prior to 2001, the year the *Agile Manifesto* [3] was published.

Hoegl and Gemuenden's work on TWQ led to many follow-up papers. Weimar et al. [28] extended the TWQ model with three additional facets in a 2017 paper: Coordination of Expertise, Value Sharing and Trust. The first factor is a management factor concerning itself with efficient resource management and thus is not considered a part of collaboration as defined in this paper (see section 2.2). They tested their modified model using questionnaire scales with 252 team members and stakeholders, showing a significant correlation between the model and team performance. Thus, Value Sharing and Trust are valuable additions to the quality model.

Lindsjorn et al. [16] studied the TWQ facets in agile teams and compared the impact of collaboration (as defined by TWQ) on agile and traditional projects, shown in figure 2.4. Again, they were able to show a strong correlation between the TWQ model and project success, as well as personal success. However, even though the two modes of teamwork show strong discrepancies in respect to the six TWQ facets, and despite agile methods emphasizing teamwork more than traditional development methods, the study did not find teamwork quality to be higher than in a similar survey on traditional teams [12]. They also found that the effects of teamwork quality is not greater in agile than in traditional teams, thus demonstrating a high need for measurable quality of collaboration for both types of teams.

Subconstruct	Traditional teamwork	Agile teamwork
Communication	More formal. Written status reports to project manager.	Less formal. Spontaneous communication (talking in the doorway, chatting, talking in front of the screen).
Coordination	Strong leadership. Project manager makes decisions; estimates, prioritizes, and delegates tasks in particular.	Not strong leadership. Self-organizing teams. The team makes decisions; estimates, prioritizes, and delegates tasks in particular.
Balance of member contribution		In cross-functional teams, it is expected that all team members contribute. Daily meetings support balance of member contribution.
Mutual Support	Hierarchical management does not facilitate mutual support among team members.	Collective code ownership, daily meetings, and retrospective meetings stimulate mutual support and collaboration.
Effort	Less focus on the team per se.	Large team focus, e.g., daily meetings. Facilitator helps protect team members from tasks outside the team.
Cohesion	Hierarchical management and more formal communication may not support cohesion.	Focus on interactions among team members, who often are physically placed together.

Figure 2.4: Comparison of TWQ facets between traditional teamwork and agile teamwork. From [16].

Chapter 3

Conceptualization

This chapter offers a structured presentation of the quality model based on literature, including all of the subdimensions and quality aspects. Since the literature is far from unequivocal on the topic of collaboration, some amount of arbitration was required for the creation of the model. Thus, this chapter will contain argumentative elements to justify the categorization and presence of the single quality aspects within the model.
The model will be presented subdimension by subdimension, with the last section of this chapter offering a final overview and visualization of the model. It should be noted, that the model synthesized from literature was only the first draft and went through slight alterations based on the findings of the subsequently conducted interview study described in chapter 4.

3.1 Methodology

In addition to the raw interview transcripts from the related work by Sasse [22] presented in chapter 2.4.1, an extensive literature analysis was conducted on the terms "collaboration" and "teamwork" in the research domain of software engineering, with the application of backward and forward snowballing for the more relevant papers. A multitude of papers was read, with a total of 14 papers being taken into consideration for the model based on the inclusion and exclusion criteria shown in figure 3.1. All types of software teams were valid objects of observation for the purposes of this model, as discussed in chapter 2.3. Key quality aspects for collaboration were extracted from all works. Aspects, which did not directly concern themselves with collaboration as defined in chapter 2.2 were excluded. For some aspects, further research in the domain of social studies and psychology was analyzed for the gathering of corresponding measurement scales. Therefore, some of the more social aspects are defined in greater detail in chapter 7.

Inclusion:
+ Describes constituents of collaboration
+ Describes constituents of teamwork

Exclusion:
- Does not have software teams as its object of observation
- Is not written in English
- Mainly concerns itself with management and efficiency factors

Figure 3.1: Inclusion and exclusion criteria for the literature analysis.

Frequently mentioned aspects, or aspects with strong evidence, were documented, combined based on similarities, and categorized into subdimensions. The subdimensions are visualized in figure 3.2 and will be explained with all of their quality aspects in this chapter. All of the subdimensions, except for *Technical Cohesion*, were also directly mentioned in literature. Due to the highly interdependent nature of socio-technical aspects [24], many of the aspects could be argued to be well positioned in multiple subdimensions. Therefore, a special focus was set on validating the positioning both in subsequent interview study (cf. chapter 4) as well as the survey (cf. chapter 5).

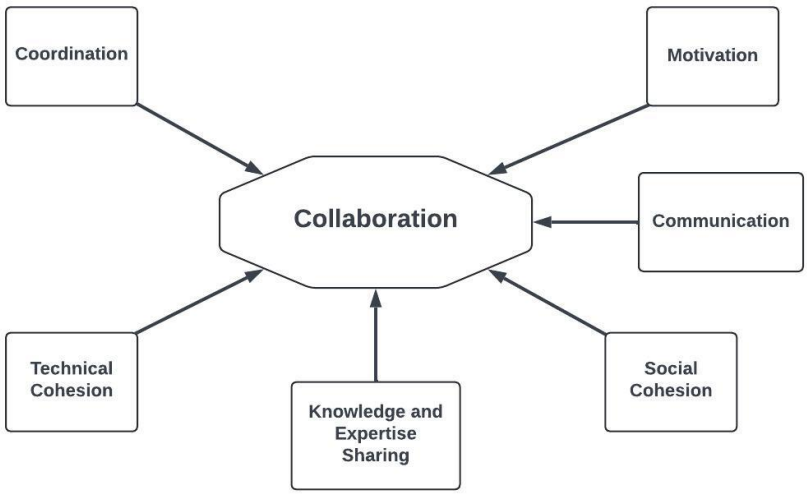


Figure 3.2: First level of the model, showing the six subdimensions based on literature analysis.

In order to avoid bloating the subsections of this chapter with repeated references, table 3.1 shows an overview of all subdimensions and aspects with the literary sources that mention the term or describe the concept as important for collaboration.

Subdimension	**Quality Aspects**
Coordination [2][12][28][29]	Software Process Framework [2][21][27] Infrastructure [2][29] Leadership [24][27]
Communication [2][12][28][27]	Information Exchange [2][12][14][4] Shared Understanding [2][12][27] Error and Conflict Resolution [24][21]
Knowledge and Expertise Sharing [2]	Peerage [1][27] Mutual Support [24][12][28]
Technical Cohesion [own]	Source Code Level Collaboration [6]
Motivation [24]	Mutual Benefit [2] Shared Values [27] Engagement [24][2]
Social Cohesion [12][28][16]	Trust [24][27][12][28] Respect [24] Sympathy [24][20] Group Membership [24][12][28][27]

Table 3.1: Summary of the model's subdimensions and quality aspects with their corresponding mentions in literature.

3.2 Coordination

The subdimension *Coordination* envelopes all collaboration mechanisms which have a coordinate or structural function for the team. These quality aspects are mostly prerequisital in nature, meaning they facilitate or support a good collaboration process, but are not directly part of the process itself. However, due to their significant impact on all collaboration that happens within the team, they serve as somewhat of a foundation for all teamwork and thus are worthy of inspection. Ideally, these aspects are already set before a project's start, but can be adjusted during the project if required. The term is explicitly mentioned multiple times in literature.

3.2.1 Software Process Framework

A software process framework, such as *Scrum*[1] (which is even explicitly defined as a *"collaboration framework"*), defines many roles, artifacts, responsibilities and events within the software development process, which all have an impact on collaboration. The roles, for example, may indicate important avenues for collaboration. The events, such as the meeting structure of the *Daily* in *Scrum*, present a foundation for other collaborative aspects such as communication. Furthermore, some studies found that progress reporting aids collaboration through synchronization of knowledge [2][27], while others found that frequent iterations increase the motivation within the team [2][21]. Common practices advocated by frameworks, such as code reviews or pair programming, aid the knowledge and expertise sharing between team members. Conclusively, attuning the software process framework to the goals and requirements for collaboration is vital to its success.

3.2.2 Infrastructure

Infrastructure describes the structural component of *Coordination* that enables collaboration through an adequate environment, especially through digital tools. Whitehead et al. [29] categorize tool support for collaboration into four broad categories, which are supplemented with modern examples:

1. *Model-based collaboration tools*, such as UML[2], allow developers to collaborate on a specific representation of the software.

2. *Process support tools* represent the development process. This includes version control systems, such as git[3], which have become arguably irreplaceable for the development of large software projects in teams.

3. *Awareness tools* aim to inform developers about ongoing work of others in order to avoid conflict. Tools for dependency management, which have been shown to improve collaboration [2], could be named as an example of awareness tools. Another popular awareness tool is the *Kanban* methodology[4].

4. *Collaboration infrastructure* tools make it possible for engineers to coordinate their work among one another. This includes tool integration such as data integration or control integration, ensuring that tools are aware of the activities of other tools and can take appropriate action.

[1] https://www.scrum.org/resources/what-scrum-module
[2] https://www.uml.org/
[3] https://git-scm.com/
[4] https://www.atlassian.com/agile/kanban

Additionally, in an increasingly distributed software development landscape, collaboration infrastructure also includes digital communication tools (such as *Slack*[5]). Mode and frequency of digital communication and meetings have been shown to have an effect on collaboration [4].

In cases of centralized teams in a classical office environment, the physical composition of the office is also a part of the *Infrastructure* aspect, since it can be more or less conducive to collaboration (i.e. through more open modern office designs).

3.2.3 Leadership

Leadership structure varies highly depending on team structure. While more traditional teams may have a clearly defined leader role, agile teams might be mostly self-led and organized. However, in both cases, there needs to be a hierarchically superior party that can make final decisions in cases of disagreement or mediate escalated conflicts, which cannot be resolved by the team members themselves. Furthermore, leadership can be an avenue for encouraging organizational politics within the team, which facilitate better collaboration (such as a culture of showing appreciation or organizing team building events) [24].

3.3 Communication

Communication is likely the most expected and self-explanatory subdimension when thinking about what contributes to good collaboration and is consequently most frequently mentioned in literature. Due to the high diversity of software team structures, the goal of this part of the literature analysis was to identify quality aspects which adequately describe the quality of communication regardless of communication mode (i.e. face-to-face meetings versus online meetings versus e-mail). Therefore, *Communication* within this model describes how information is processed, exchanged and shared between team members, with *Error and Conflict Resolution* being included as a *"special communication need"* [21].

3.3.1 Shared Understanding

Shared Understanding is an established term and an ongoingly researched concept explicitly mentioned in multiple papers. In short, *Shared Understanding* can be defined as the absence of misunderstandings. Batra et al. [2] note that, for example, a mismatch in expertise can result in misunderstandings. They state the following:

[5]https://slack.com/intl/en-gb

"The way to eliminate misunderstandings and uncertainties about the software product requires that the team members [...] communicate with each other to reach a shared understanding."

Therefore, *Shared Understanding* is both an important requirement for high quality communication, while also being the result of high quality communication. This clearly justifies the positioning of the quality aspect, while also demonstrating a strong interaction with the superior subdimension of communication. The existence of misunderstandings between team members and stakeholders of a project has been stated as a main cause of project failure [28].

3.3.2 Information Exchange

Information Exchange describes the raw part of communication that involves information being transmitted from one party to another. This is likely the most substantial aspect of communication which is present in all forms of communication, regardless of global and temporal distribution of the individual team members. *Information Exchange* revolves around the process of information, which has previously been externalized by a team member, being internalized by a different team member. It includes direct communication, such as face-to-face meetings or phone calls, but also includes indirect communication, such as e-mails and all types of documentation.

3.3.3 Error and Conflict Resolution

Errors and conflicts are unavoidable events in all large scale software teams. When encountering a committed error, it can be a very sensitive issue which needs to be handled correctly. Practicing developers in an interview study [24] noted, that it is important to ask the right questions when somebody made an error, otherwise team members might be hesitant to admit mistakes in the future. Instead of looking for someone to blame and asking "Who is responsible for this error?" it is more constructive and socially advantageous to ask "How did the error happen and how can we avoid similar mistakes in the future?". Therefore, communication is at the center of *Error and Conflict Resolution*. Similarly, Paasivaara et al. [21] describe problem-solving practices as an *"extremely important communication need"*. Therefore, high quality communication needs to include a constructive and open error culture, justifying the positioning within the communication subdimension.

3.4 Knowledge and Expertise Sharing

The sharing of knowledge and expertise in the domain of software engineering goes past mere information exchange. Due to many popular teamwork practices in modern software teams (such as pair programming or code reviews), the quality model deserves an explicit subdimension for all these mechanisms.

3.4.1 Peerage

Peerage, as used by Augustin et al. [1], describes important interactions between peers, both on a social and a technical level. The authors first identified this aspect as a key difference between early 2000s organizational software development and successful online open source projects. It was concluded to be a reason for why collaboration in these online projects has higher quality and produces superior software compared to commercially available alternatives. The mechanisms have been widely recognized since then as a unique need for collaboration in software teams.

Traditional office work can often include high pressure, competitive environments, where the employees compete for recognition and praise from their superiors. While a competitive environment could boost motivation for individual tasks [12], it also restricts the mutual support and expertise sharing that is vital for collaborating and growing as a software team. Similarly, the validation in software teams is also more effective when it comes from people of similar or superior expertise in terms of programming. A manager with little programming knowledge cannot validate the quality of a developers work as well as a fellow developer. This is one of the reasons *code reviews* have been established as a standard collaboration practice in modern software teams [27]. Additionally, when the dynamic between peers is ideal, the acknowledgment of one's work should also be sought out from peers, rather than from one's boss. Augustin et al. [1] noted that one of the motivations of the developers in successful open source projects was to impress other skilled developers, which increased the quality of their work. Lastly, the education and training within highly technical teams teams also happens between peers. An example for peer education in software teams is high level documentation, or the well-known practice of *pair programming*, which includes a constantly high transfer of knowledge and expertise between developers.

3.4.2 Mutual Support

According to Hoegl and Gemuenden [12], mutual support is the willingness between team members to help and support each other in carrying out their tasks. In software teams, this often includes sharing one's knowledge or expertise with other developers. It is very important for mutual support, that the team atmosphere is not too competitive, especially in regards to seeking praise from a superior. Otherwise, the developers could be much less likely to help and support each other.

As noted by a practicing developer in an interview study [24], software teams can include ego-driven personality types, who may think they are infallible. In these cases, it is almost more important that the team member is willing to let themselves be supported. This is also an important part of high quality *Mutual Support* and likely requires good levels of trust between team members.

3.5 Technical Cohesion

Technical Cohesion describes collaboration on an exclusively technical level, such as how the developers are interacting with each other on a source code level. This subdimension was inspired by the work of Caglayan et al. [6], in which they study the effect of developer collaboration activity (purely technical) on software quality in two large scale projects. The authors studied the impact of collaboration networks on defect proneness in software projects. They presented collaboration networks as a valuable tool for measuring technical collaboration in software teams through the metrics of the graph itself.

At first, it may seem redundant to explicitly look at technical collaboration, when it could be argued to be the result of the quality of collaboration as a whole. However, the authors also cite, from a previous work, that it has been empirically observed, that the collaboration structure on a purely technical level may be significantly different from the collaboration structure on an organizational level. Previous research concluded *"dramatic differences between the organizational team structure and the actual code-level collaboration structure."* [6]. Therefore, it was considered worthy of separate inspection within the model.

3.5.1 Source Code Level Collaboration

Source Code Level Collaboration describes how developers interact purely on a code base level. It is a dynamic that can automatically be generated from many repository systems (for example *git*) through *collaboration networks*.

Caglayan et al. [6] offer a brief introduction into collaboration networks in their previously mentioned paper. An example of a very simple collaboration network is provided in figure 3.3. In the network graph, developers represent

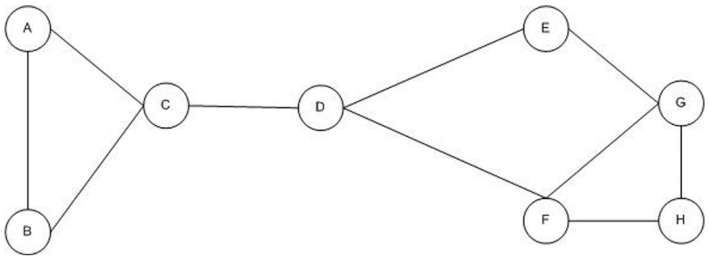


Figure 3.3: Sample collaboration network from [6].

a node. If developers collaborated on at least one software module, an edge is drawn between their nodes. Collaboration, within this context, is exemplified by the authors:

Example: *"[...] two testers have originated two issues. A developer changed the same software module related to the issue."*

In this case, it is assumed implicitly that the two developers have collaborated. This assumption is supported by developer interviews conducted in three open source software projects by Meneely et al. [19]. Many rich metrics in regards to technical collaboration can be extracted from the collaboration network through the metrics of the graph itself, such as centrality, degree and betweenness, making it a very easily measurable quality aspect. In our example we can already identify some potentially problematic dynamics from the graph. We can see that there are two sub-groups in the graph, namely A,B,C and D,E,F,G,H. Between these groups, collaboration only happens between developer C and D, who form a bridge. If either developer becomes unavailable, there may be no avenues for cross-collaboration between the developers of the two sub-groups. Of course, this is a simplified example and real collaboration networks (see figure 3.4) are a lot more complex, which is why they can only be analyzed through metrics instead of manual inspection.

3.6 Motivation

Motivation is likely the most individual-centered subdimension. being more of a social aspect, it is rarely mentioned in literature, however based on the opinion of practicing software developers [24], it is frequently mentioned as an important requirement for a good collaborative process. Therefore, when looking at how to achieve high quality collaboration, it is relevant to analyze how to increase the team members' motivation for collaboration.

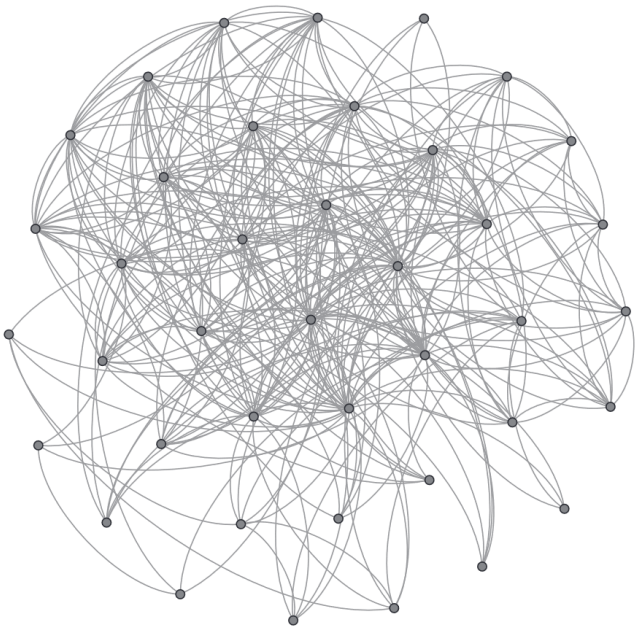


Figure 3.4: Collaboration network extracted from a real world project, taken from [6].

3.6.1 Mutual Benefit

Mutual Benefit boosts motivation for collaboration through personal enrichment of the collaborating parties. This can come in many shapes. Some team members may feel they already benefit enough through the financial compensation they receive for their work. Others might take this for granted and might require additional benefits, such as being able to learn something new through the collaboration. For example, by collaborating with technically skilled peers or working on advanced technologies which offer a learning experience for the developer. What is most important for *Mutual Benefit* is the personal perception of each developer that they get something adequate from their work.

3.6.2 Shared Values

A congruent perception between team members of what is important within the team is a vital factor for a good collaborative process. This may include both moral and technical values. For example, if there is a strong discrepancy in degree of perfectionism between team members, one party might be left unsatisfied with the result of the work, while the other might feel that they are wasting too much time for marginal increases in quality.

An example for the difference in technical values is the following: In a project developer A and developer B are part of a software team. Developer A values code simplicity and readability very highly. Meanwhile, Developer B values code efficiency very highly and is ready to heavily increase the complexity of a module for a minor performance increase. These two developers have conflicting values, which may result in decreased motivation for collaboration. An example for a difference in moral values would be the individual's opinion on the team's ideals in regards to social interactions. While some developers of a team might like frequent social interactions and might desire more personal exchange, other, more solitary, developers could be annoyed by frequent meetings or planned social activities. This will always leave one fraction unsatisfied, resulting in a negative impact on motivation.

Aligning these values and compromising on them within a team can increase the motivation for collaboration [28].

3.6.3 Engagement

Engagement envelopes all personal factors which contribute to an increase in involvement or commitment to the collaboration. While *Engagement* could technically be measured by tracking the productivity of individual developers over time, the contributing factors are often highly individualized: *"Will the project be a success? Do I think the project has a valuable purpose or mission? Do I like the people I'm working with? Is the work itself interesting? Will my work be acknowledged?"*. This makes engagement another highly subjective quality aspect which can likely only be measured by capturing the personal perception of a team member.

Batra et al. [2] note that involvement and commitment increase the longer a developer is part of a team and the more cumulative collaborative successes take place. Conclusively, it is both a contributor to high quality collaboration, while also being strengthened by it. It probably also has high levels of interaction with the perception of *Group Membership* (cf. chapter 3.7.4).

3.7 Social Cohesion

Social Cohesion describes general social aspects which are important for the quality of collaboration in all types of teams. Consequently, they are also highly important for software teams. The term is vaguely mentioned as "cohesion" in some literature [12], but requires more detailed inspection in order to methodically increase its quality. Social cohesion within a team can improve communication, increase motivation and reduce conflicts [28].

However, it is likely the least studied subdimension within software teams (though heavily studied in general social research). Consequently, many of the aspects in this subdimension are mostly sourced from interviews of the personal accounts of practicing software developers without substantial backing from literature relevant to software engineering.

3.7.1 Trust

Trust between team members is frequently named as an absolute prerequisite for collaboration [24] and is possibly the most important contributor to *Social Cohesion*. Mayer et al. [17] define trust in an organizational context as follows:

> *"the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control the other party."*

Trust within a team is a supporting mechanism for teamwork, influencing many processes such as the willingness to share information and expertise, communicate openly, give honest feedback or admit errors [28][24]. Salas et al. [22] define mutual trust as follows:

> *"The shared belief that team members will perform their roles and protect the interests of their teammates"*

Conclusively, trust in a fellow team member can mean many things: Trust in their ability, trust in their reliability, or the trust in their good intentions and best effort given, without the need for monitoring that team member. Examples for where these forms of trust find direct application in software engineering are code reviews and dependency management.

3.7.2 Respect

Respect is an aspect that was deemed very important by practicing software developers [24], but was not explicitly mentioned in relevant literature. Thus, it is an aspect which is possibly overlooked in software engineering literature. While some might take it for granted, a lack of respect within a team was stated as a reason for severe issues with collaboration. Especially in cases of superiority in skill or hierarchy, it is most important that the members of a team talk to each other as equals. A participant in the interview study by Sasse et al. [24] mentioned an anecdote, in which a technically excellent developer constantly belittled the work of a team member. The belittled team member slowly started refusing to communicate and eventually refused to partake in meetings with the belittling team member. The interviewed

developer stated that they found the actions of the belittled team member reasonable and would also refuse to work with someone who did not respect them. The stated experiences of software developers could suggest that a lack of respect impacts communication, lessens motivation and severely damages the social cohesion of a team.

3.7.3 Sympathy

Sympathy describes the congruence of understanding and feeling between team members. It is likely the most personal and hard to influence quality aspect of the entire model, as it revolves around the personal relationships of the individual team members. Mullen et al. [20] stated an *"interpersonal attraction of team members"* being important for (social) cohesion. However, similarly to *Respect*, it was mostly mentioned as important by practicing software developers [24] and rarely explicitly mentioned in relevant literature. Developers stated that working together is more enjoyable if you personally like the people you work with. Higher sympathy could also lead to people viewing their team as their social circle which could lead to them remaining in the same collaborative constellations for longer, which in turn has a positive impact on involvement and commitment [2]. It is often attempted to increase the sympathy between team members by engaging them in social, non work-related activities with each other (team building events) [24].

3.7.4 Group Membership

An adequate feeling of being a part of the team is sometimes defined as the cohesion of the team itself [28]. It envelopes abstract concepts such as "team spirit" [12] or a sense of belonging. Mullen et al. [20] explicitly state that *"an adequate feeling of togetherness and belonging is needed to achieve high quality collaboration"* . It is in many ways similar to *Sympathy*, however *Group Membership* exists in relation to the entire team, while sympathy can be more or less strong between individual members of a team. High perception of group membership leads to higher motivation to maintain the team [12] and higher team orientation [27]. It is likely a concept that is enabled through continuous collaboration and shared successes in a team, as well as high quality of the other social aspects (*Trust, Respect and Sympathy*). It is also attempted to be increased through team building events [24].

3.8 Overview and Further Categorization

The first draft of the model was concluded after many decisions about the inclusion and categorization of the quality aspects had to be made. These were sometimes partial arbitration due to the ambiguity in literature. Therefore, there was a high need for validation in regards to these decisions. In order to adequately explain the model to potential interview candidates, a visualization of the model was prepared, which can be seen in figure 3.5. In order to make the visualization more digestible for first time viewers, the model was subdivided into further categories. On the horizontal axis, the aspects were subdivided into a technical and social plane (with aspects in the middle being defined as socio-technical). Furthermore, on a vertical axis the aspects were subdivided into aspects prerequisital for collaboration (i.e. *Coordination* and *Motivation*) and substantial collaborative aspects (i.e. *Technical Cohesion* and *Knowledge and Expertise Sharing*). *Communication* falls into both the prerequisital and substantial plane for collaboration. It is notable, that while it was highly anticipated before the literature review, the concept of "documentation" was, to the best of knowledge, not used in any literature on what constitutes high quality collaboration in software teams. While it is indirectly included in the quality aspects *Peerage* and *Information Exchange*, it was not directly included as a quality aspect due to a lack of a literature reference.

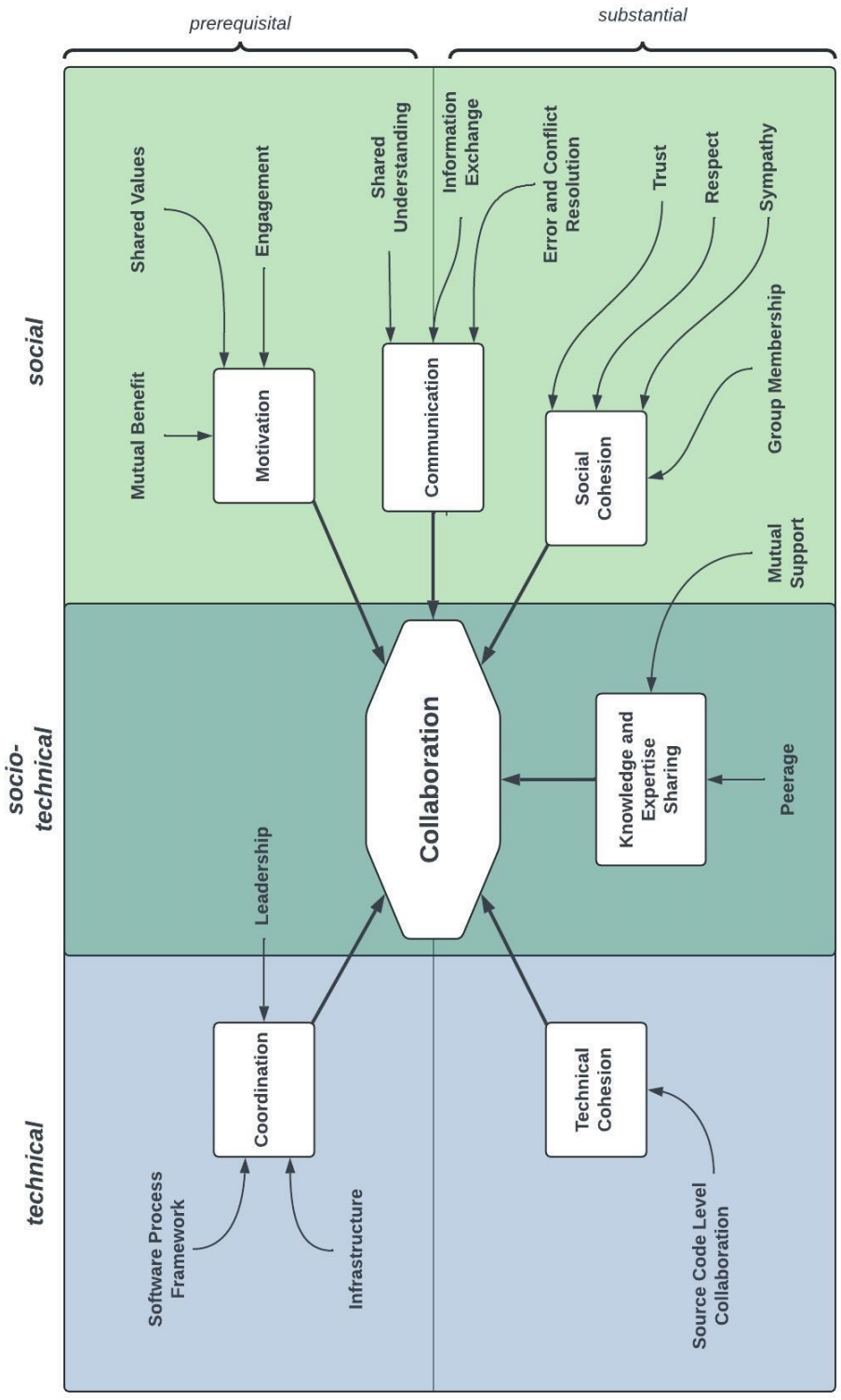


Figure 3.5: First draft of the quality model, based on literature. The visualization is supplemented by categorization into more or less social, as well as prerequisital and substantial aspects.

Chapter 4

Interview study

In order to get initial feedback on the model which, so far, has mostly been based on literature, an interview study was planned in order to validate the model and possibly gain new insights. The main focus of the interviews was to validate the model for *completeness* in order to ensure that no major aspects were forgotten and that no aspects were completely out of place. The secondary focus was on validating the correct positioning of the quality aspects within the subdimensions. Even if no major disagreements were found, the interviews could serve to confirm the various decisions which were made as part of the conceptualization and thus strengthen the model for the subsequent survey study. The interviews were conducted in German. This chapter outlines the details of the interview structure, their participants and the findings that led to eventual changes of the model.

4.1 Study Design

Two research questions were formulated to aid the design of the interview structure:

1. Which quality aspects or subdimensions, if any, are missing from the model?

2. Which quality aspects could be better positioned under a different subdimension?

As the main purpose of the study was to validate the model for completeness, a structured interview would certainly have been unsuitable due to the low degrees of freedom in the possible answers of the participants. However, since the exploration of the interviewees should take place within the context of the quality model, some structure had to be given, including a fundamental explanation of the model and its purpose. However, only the subdimensions were given and explained to the participants. Revealing

the individual quality aspects could have conditioned the participants too much and could have prevented them from thinking about possible quality aspects based on their professional experience. Therefore, a semi-structured interview guide was designed, where relevant tangents and anecdotes of the interviewees were constantly encouraged and welcomed. Possible follow-up questions were prepared in case the participant answered questions in a certain way. In order to best answer the research questions the following interview guide was created:

1. **Welcoming**: This section mainly served the purpose of offering a brief overview of the interview to the participant and of building initial rapport.
    - The participant was greeted and an appreciation for their time and participation was expressed.
    - An outline of the topic was given.
    - A rough estimate for the time and structure of the interview was given.
    - The pre-signed data protection agreement was briefly recapped and the participant was informed that the recording will now be started.
2. **Initial personal questions:** To gather some personal data and encourage active thought of the participant, some personal questions were asked before transitioning into the main topic.
    - Question: *"How old are you and how long have you been working in software development?"*
    - Question: *"In which contexts and constellations have you worked in teams to create software?"*
    - Question: *"How important was the quality of collaboration in these contexts?"*
    - Question: *"Did your experiences in any of those teams significantly differ in terms of collaboration?"*
        - If *"yes"*: *"How did the differences impact the team and how were the deficiencies handled?"*
    - Question *"Do you know what a quality model is?"*
        - If *"yes"*: Let participant attempt to explain it and correct possibly inaccuracies.
        - If *"no"*: Briefly explain quality models.

3. **Model disclaimer:** In this section, the quality model for collaboration was superficially explained in order to prepare the participant for the subsequent exploratory section.

    - The participant was informed that a brief overview of the quality model would be given to them and that the interviewer will just go over the subdimensions, talking about the individual quality aspects at a later point. It was assured that the participant was free to ask questions at any time.
    - The participant was explicitly encouraged to give as much feedback and critique of the model as possible.
    - A visualization of the model, including the six subdimensions without quality aspects, was shown and explained to the participant.

4. **Key Questions:** During this main part of the interview, the participant was mostly encouraged to think about their own experiences, recall anecdotes, and explore as much as possible. For each subdimension (i.e. *Communication*), the participant was guided through the following process:

    - Question: *"What comes to mind when you think about «subdimension» in regards to collaboration?"*
        - Possible follow-up or nudge: *"What constitutes or contributes to the quality of «subdimension» in software teams?"*
    - Next, the researched quality aspects of the subdimension were revealed and compared with the statements of the interviewee. Agreements were noted, disagreements were discussed.
    - Question: *"Do you think that the researched quality aspects are accurately positioned within «subdimension», or would some be more fitting in another subdimension?"*
    - Question: *"Overall, can you think of any further important quality aspects that contribute to «subdimension» that have not been discussed so far?"*

5. **Closing Question:** After the model was discussed in its entirety with the participant, they were given a last chance to think of anything that contributes to collaboration.

    - Question: *"In regards to everything we have talked about, is there anything else you would change or add to the model?"*

6. **Ending:** the ending section served the purpose of concluding the interview and reminding the participants about the conditions of the interview.

- The participant was again thanked for their participation.
- The participant was reminded that the gathered data would be anonymized and published as part of a master's thesis.
- The participant was informed that the recording would be stopped at that moment.

A pilot interview was conducted with an academic researcher, before the interview guideline was finalized and applied for the interviews on the real participants.

4.2 Participants

The interview was conducted on four participants (excluding the pilot participant) who were all practicing software developers with multi-year experience of varying lengths and domains. Since the main focus of the interview study was to identify bigger potential oversights or inaccuracies, a small participant group was deemed sufficient considering the group had adequate diversity in work experience. Some demographics of the participants are outlined in table 4.1. The most valuable diversity of the

Number	Age	Experience	Occupation
P1	28	2.5 years	Junior Software Developer
P2	28	3 years	Software Developer/Researcher
P3	28	7 years	Senior Software Developer
P4	60	31 years	Enterprise Architect

Table 4.1: Interview study participants.

participants likely lay in the varying types of their organization and the resulting diverse experience of software team constellations. Participant P1 was a team member of a regular software team in a small size software development company. Participant P2 was working as a software developer and researcher for a health related state institution, which means he was simultaneously part of many different, often cross-functional teams. Participant P3 was working for a large German automobile manufacturer, which resulted in many different types of software team constellations, both fully local software teams as well as team constellations with exceptionally large geographical distance (i.e. a team where P3 was the only member working from Germany, while the rest of the team was based in India). The last participant, P4, had multi-decade experience as a software developer, further experience as a software architect and was working as an enterprise architect during the time of the interview. He worked fully remote for a very large international conglomerate, where his team was based in four different countries, making remote work the only possible mode of collaboration.

4.3 Main Findings and Revision of the Model

All participants were able to contribute valuable, experience-based anecdotes, which strengthened existing quality aspects, broadened their definitions and even prompted the addition of further quality aspects. All participants managed to identify important quality aspects for each of the subdimensions. Sometimes, the participants approached a subdimension very similarly to the research-based model, listing almost identical quality aspects. In other cases, they approached a subdimension from a different perspective, leading to different quality aspects, while still being unifiable with the researched model.

4.3.1 Motivation

Motivation was likely the subdimension the participants were most eager to talk about. P2 defined *Motivation* by subdividing it into intrinsic and extrinsic motivation. This is an interesting and different approach, which is still congruent to the model, where *Engagement* and *Shared Values* are covering intrinsic motivational factors, while *Mutual Benefit* describes all external contributors to motivation. The participant noted that the revealed aspects were congruent with his view on motivation. Both participants P3 and P4 mentioned *appreciation* being important for motivation. P3 referred to the companies' appreciation for the IT division as a whole (he noted that they are often secondary to the engineering division of the automobile manufacturer). P4 referred to appreciation for one's personal work. This was an important addition, however it was deemed a contributor to the existing quality aspect of *Engagement*, broadening its definition. Furthermore, the concept of appreciation is further included in the *Peerage/Peer Validation* quality aspect and partially covered by the *Leadership* aspect (organizational politics being brought into teams, including organizational appreciation). Both participants also mentioned a large degree of freedom in their own work and design decisions as motivating. However, this can be a personal preference with some developers possibly appreciating more clearly defined constraints. Thus, it was also interpreted as a contributing factor for *Engagement*. P3 also expressed the opinion, that money stops being a strong motivator after a certain point which is quickly achieved in software development, thus needing further motivators, such as being able to work with new technologies or knowing that the software product will be used by, and help, many people.

4.3.2 Coordination

Coordination was the subdimension where the participants' own thoughts were most identical with the existing model. The participants all named quality aspects which were researched, with P2 even naming all three of the existing quality aspects. Nevertheless, there were many valuable additions and anecdotes from the participants. P3 and P4 noted that the term *Infrastructure* can be misleading, as it can be interpreted under more technical meanings in the domain of software engineering. P3 mentioned that it is important for the infrastructure to allow platforms, through which the developers can freely choose the software tools they wish to use. He also noted the overall structuring of the teams within the organization as part of coordination, which is a valid point that was not included due to the model focusing on intra-team collaboration. When talking about *Leadership*, he strongly expressed his opinion that putting the organizational politics over everything else leads to poor coordination. It was also mentioned that the organizational mentality in regards to work from home options influences the quality of collaboration, which would be included under *Leadership*.

4.3.3 Communication

When talking about *Communication*, the participants mostly had different approaches for defining the subdimension. Multiple participants talked about quality of *Communication* being influenced by the type of communication channel used, which is a valid argument but is not included in the model due to the reasoning presented in chapter 2.3. All four participants talked about *openness* being important for *Communication*, with P2 and P3 explicitly mentioning an open error culture and blame-free environments. An open communication is indeed important. However, the degree of openness in communication is likely a result of quality aspects for *Social Cohesion*, most importantly *Trust* and *Respect*. Furthermore, the result of the openness of communication can be indirectly measured by the three existing quality aspects in the *Communication* subdimension. Therefore, it was not explicitly added as a quality aspect.

P3 contributed an anecdote about the importance of a balance in communication. In a previous software team, he was the only developer working from Germany in a team that was based in India. As a result, he often felt excluded in the information exchange of the group, suspecting that much information was exchanged informally outside of team meetings. The metric which will be suggested for the *Information Exchange* quality aspect covers this by analyzing communication network metrics such as centrality (cf. chapter 7), which would be well suited to identify the problematic dynamic described in the anecdote. P4 expressed the opinion,

that *Communication* and its quality aspects are too complex and need to be further subdivided into a "fourth level". This is a valid concern, considering the level of detail you can go to when analyzing communication. However, in order to keep the model balanced and comprehensive, the suggestion was not included in any revisions.

4.3.4 Knowledge and Expertise Sharing/Peerage

When explaining *Peerage* to the participants, it quickly became clear, that the quality aspect contains too many distinct and important facets, which were often individually referred to by the participants as part of *Knowledge and Expertise Sharing*. Participants P2 and P3 named "documentation" as part of *Knowledge and Expertise Sharing*, with P3 contributing an interesting anecdote about a company which tried to boost collaboration by monetarily incentivizing high quality documentation. They mostly referred to high level documentation (i.e. describing the structure of whole software modules as opposed to low level code documentation). This contributed to the assumption, that the education and training between peers is too hidden within the *Peerage* quality aspect and likely deserves to be its own aspect within the model. This assumption was further confirmed by participants P1 and P4, who additionally mentioned practices from their experience which consisted of team members giving small presentations or lectures to the rest of the team about new or interesting technologies they are currently working on. Similarly, the participants frequently mentioned the practices of code reviews and pair programming as parts of *Knowledge and Expertise Sharing*, which led to a large restructuring of the subdimension visualized in figure 4.1. Mutual support was mentioned by P2, who noted that an absence of competition and pressure is good for both mutual support and social cohesion.

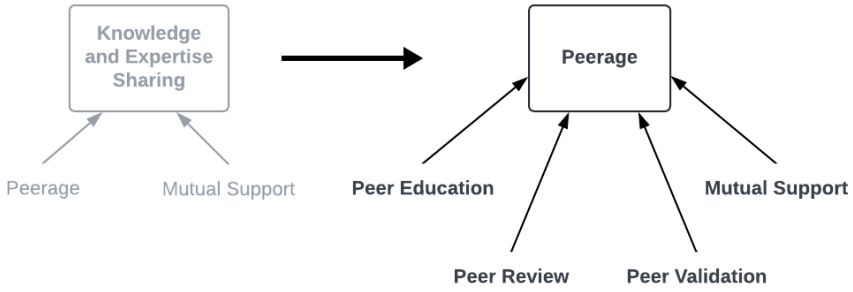


Figure 4.1: *Knowledge and Expertise Sharing* turns into the *Peerage* subdimension, allowing for more detailed inspection of quality aspects.

4.3.5 Technical Cohesion

Technical Cohesion was a subdimension originally only consisting of a single quality aspect. It was extended as a result of multiple participants describing a concept which could be summarized under the term *Technical Consistency*. P2 talked about how it is important *"how coherent and unified a project is being worked on"* and that there has to be a *"consistency of technologies"*. This could likely be the result of the participant being a member of multiple, cross-functional teams. If the team members use different technologies (i.e. different frameworks or different programming languages), it is harder for the developers to collaborate in a technically cohesive manner. P3 also said, that it is important to agree on a technical basis within a team (listing programming language, platform, provider and server as examples). P4 echoed a similar sentiment, specifically naming the example that collaborating parties should use the same IDE (Integrated Development Environment) if they work with the same language and that discrepancies have led to problems in his past experiences.

It was previously assumed that, ideally, a perfectly unified and consistent technical environment is dictated by the *Infrastructure* quality aspect of the *Coordination* subdimension. However, this does not seem to be the case in some software organizations (partially due to developers sometimes being members of multiple teams at once). It is also in slight conflict with the statement made by P3 on the topic of infrastructure, in which he described a high degree of freedom when choosing software tools from the infrastructure platform as important. Furthermore, the idea of technical consistency also describes a facet which has not yet been considered and is likely important for collaboration: Coding conventions. Inconsistent coding conventions can complicate effective collaboration, especially in combination with version control systems. Therefore, *Technical Consistency* was included as an additional quality aspect in *Technical Cohesion*.

No participant named a quality aspect similar to *Source Code Level Collaboration*, however, all participants expressed strong interest in the concept after having it explained to them, with some even mentioning that they will try to implement ways to measure this in their own software teams.

4.3.6 Social Cohesion

P2 said, that it is important for everyone in a team to be hierarchically on the same level. While this is something that cannot be guaranteed in all types of teams, treating each other as individuals on the same level is something that is described within the *Respect* quality aspect. Furthermore he noted that having no competition or pressure within the team, as well as good approaches to resolving conflicts, contributes to *Social Cohesion*. This is congruent to the quality aspects *Mutual Support* and *Error and Conflict Resolution*, which were not part of *Social Cohesion*. Retrospectively, the participant agreed that the quality aspects are likely better positioned in their respective subdimensions, however noting strong interdependencies. Lastly, he noted that shared rituals (such as having lunch together) and team building activities contribute to the feeling of being a team. These were noted as contributing factors to the quality aspect *Group Membership*.

P3 noted that the most important factor for *Social Cohesion* is having a shared enemy or goal, and furthermore having a shared understanding of how to achieve this goal. These thoughts are included in the quality aspects *Shared Values* and *Shared Understanding*. The participant eventually agreed with their placements, naming these more examples of interdependencies of quality aspects and other subdimensions. P3 also mentioned an anecdote where an attempt was made to include the superior of the team division within the team. He explained that this was a failure due to the vast hierarchical differences between the superior and the members. When discussions arised, many team members did not dare contradict the opinion of the superior. The participant noted that this experiment destroyed much of the social dynamic of the team.

P4 explained that the social cohesion of his teams was always better when they were together in person and that there is almost no social cohesion within his current team, which is fully globally distributed and socio-culturally distanced. He mentioned that attempts were made to create social cohesion through team building activities, but these were apparently not fruitful. This is another example of sociocultural and global distance negatively impacting the social cohesion of teams.

4.3.7 Further Findings and Adjusted Visualization

All participants frequently noted interdependencies between both quality aspects and subdimensions (for example good *Error and Conflict Resolution* being important for *Social Cohesion*), however, in the end, no explicit suggestions were made to move any of the quality aspects into a different subdimension. P3 noted that he definitely wanted to see the aspect of "documentation" included as a quality aspect in either *Knowledge and Expertise Sharing* or *Communication*. This demand is considered satisfied by explicitly including *Peer Education* as a quality aspect, covering the high level documentation, while *Information Exchange* also considers various types of documents.

As expected, multiple participants mentioned high geographical or sociocultural distances as counterproductive for collaboration, for example stating that face-to-face meetings are better for communication than online and hybrid meetings, or in-office-work and shared rituals (such as having lunch together) being beneficial for the social cohesion of a team. While all of these comments were reasonable and stem from real world experience, these factors could not be included in the model due to choosing the approach of designing it in a way that allows application regardless of temporal, geographical and sociocultural distances (as argued in chapter 2.3).

While the participants showed consideration for the visualization's subdivision into technical and social planes and contextualized some quality aspects under these categories, no participant engaged with the subdivision into prerequisital and substantial aspects. Therefore the vertical subdivision was scrapped from the model. All of the participant feedback led to a revised visualization of the quality model, shown in figure 4.2, which was then used for the subsequent survey study.

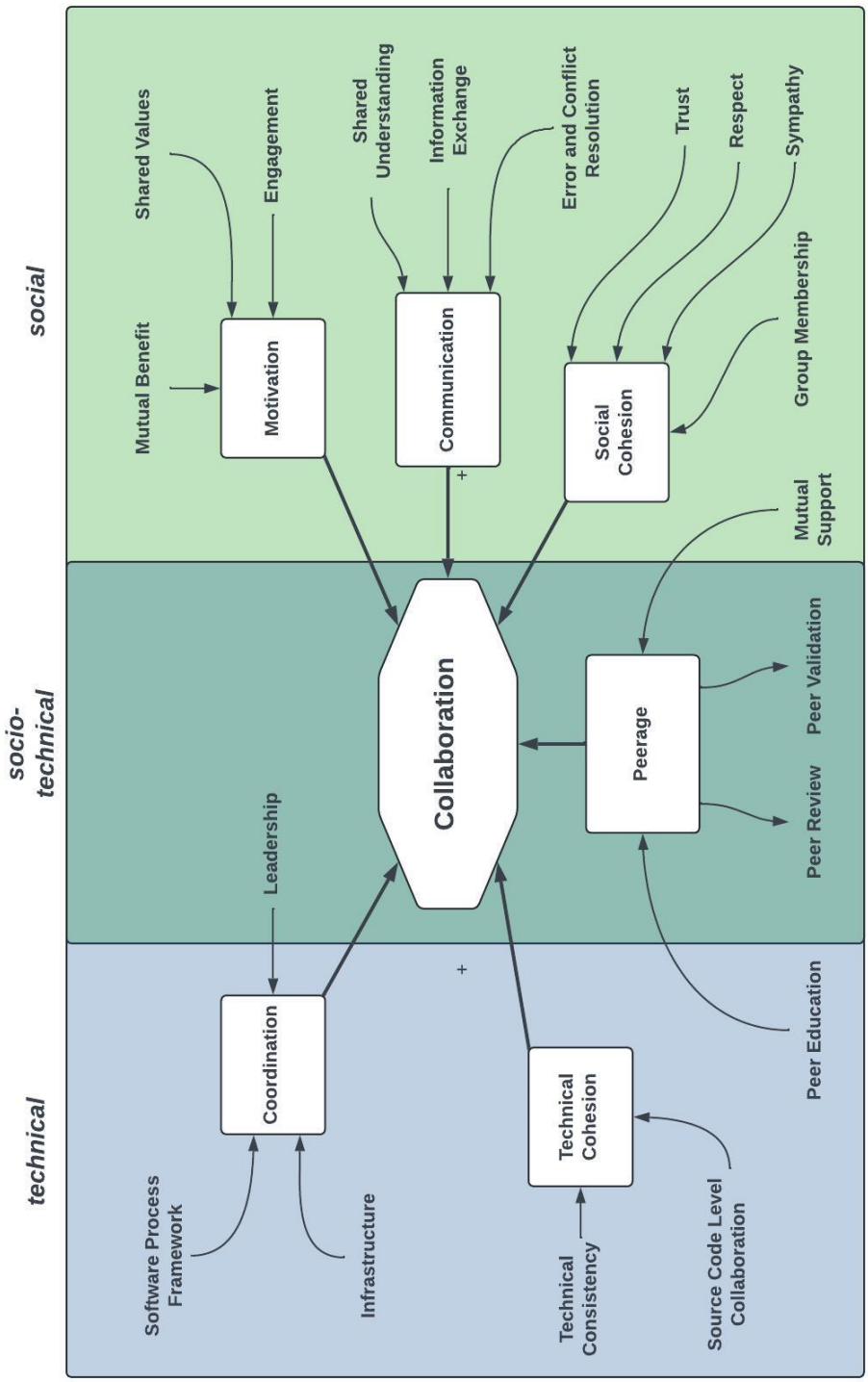


Figure 4.2: Visualization of the quality model for collaboration, based on literature and interview feedback.

Chapter 5

Survey

The model was validated for completeness in the interview study, however due to the relatively low amount of participants there was no quantifiable data that could be gained from the study. In order to further validate the model in a quantifiable way, an online survey study was conducted with a resulting total of 58 validly completed response sets. This chapter describes the study design as well as the demographics of the participants. Lastly, the results of the survey are presented through visualization and commentary.

5.1 Study Design

The main focus of the study was to evaluate the importance of the existing quality aspects in a quantifiable way. This could lead to insights about certain quality aspects being indispensable, while others could possibly be left out of the model. The latter aspect is something that could not be determined from the interview study at all, since the participants were only asked if any quality aspects were missing from the model, not if any of them were unnecessary. As a secondary objective and as an additional layer of validation to the interview study, the participants were also asked if they found any quality aspects non-optimally positioned or missing entirely from the model. However, those were optional questions which did not have to be answered in order to progress through the survey. The design of the study was guided by two further research questions:

1. How do the quality aspects rank in terms of importance for collaboration?

    (a) Which quality aspects, if any, are deemed indispensable for collaboration?

    (b) Which quality aspects, if any, are deemed unimportant enough to be left out of the model?

2. Are there any subdimensions or quality aspects missing or non-optimally positioned in the model?

Similar to the interview study, the participants needed to make their evaluations within the context of the quality model. Therefore, it was also necessary to explain the model to them before they could express their opinion in an informed way. For example, when asking people about the quality aspects of a subdimension and whether they would be better positioned in a different subdimension, they first have to roughly understand all of the subdimensions as defined within the context of the model. This was a concern for the full completion of the study, as people can be very quick to abort their participation in an online survey if the content seems too complicated, wordy, or otherwise bothersome. Therefore a special emphasis was placed on keeping the necessary initial explanation of the model as brief and simple as possible. For the same reason, the entire survey was planned to only take an average of 15 to 20 minutes in total. The survey was designed as follows:

5.1.1 Personal Questions and Explanation

The first part of the survey served the purpose of informing the participants about the conditions of the study and to very briefly explain and outline the quality model to them. In addition, the participants were asked about their age and length of work experience. This data was gathered to find possible correlations between the age or work experience of a participant and their degree of opinionation, as well as their priorities in regards to what constitutes high quality collaboration. The initial explanation only consisted of definitions of the concepts “collaboration” and “quality models”, as well as brief, one-to-two sentence, descriptions of each subdimension with a visualization of the quality model. As explained, this was kept as concise as possible in order to avoid people closing the survey after being confronted with too much information at once.

5.1.2 Rating the Quality Aspects

In the main section of the survey, the participants were asked to rate the quality aspects in terms of importance for collaboration. In each step, a closer look was taken at the subdimensions one at a time. Each of their quality aspects was explained to the participants, with the explanation of the corresponding subdimension and the visualization of the model (cf. figure 4.2) being left on the page in order to offer a reminder and help for contextualization to the participants. The participants were asked to rate the importance of the quality aspects on a 10-point-scale, with “1” meaning entirely unimportant and “10” meaning extremely important. Rating each aspect was mandatory in order to progress through the study. Furthermore,

in each step the participants were asked if any of the explained quality aspects would be better positioned within a different subdimension of the model. This was an optional question. An example of this question group (for the subdimension of *Coordination*) can be seen in figure 5.1.

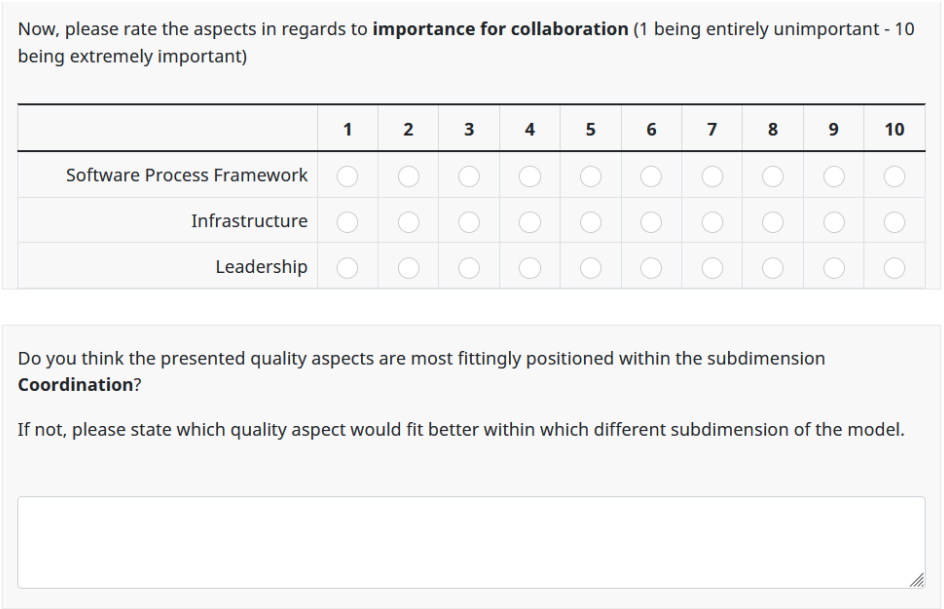


Figure 5.1: Question group participants answered for every subdimension after being given an explanation of its quality aspects.

A very important concern for the results of the study was the fact, that the "importance" of an aspect is something that is likely rated in relation to other aspects. If all participants were exposed to the quality aspects of *Coordination* first, they would likely be subject to an anchoring bias in their subsequent ratings, rating other quality aspects in relation to the first ratings they gave (more or less important). For example, if a participant rated the quality aspects of *Coordination* with an average of "8", the average of all their ratings would be more likely to be close to "8". If another participant deemed the quality aspects of *Coordination* less important, with an average of "5", the average of all their ratings would likely be lower than the ratings of the first participant. Therefore, some degree of counterbalancing was mandatory in order to avoid severe ordering effects of the survey results. Since the study population was expected to be sufficiently large, the counterbalancing was implemented by fully randomizing the order of the middle part of the survey. Meaning, each participant rated the quality aspects in a different order, therefore negating systematic ordering effects.

5.1.3 Closing Opinions and Comments

In the last section of the survey, the participants were given a final chance to give any opinion or comment on the model. This is similar to the final section of the interview study, however it was even more important for the survey, since there was no opportunity to stimulate potential ideas and comments through dialogue, instead being the result of the internal thoughts the participant had during the survey. In three optional questions, the participants were given the chance to submit free-text answers under a full visualization of the model:

1. In your opinion, is there a larger **subdimension** (big rectangles) missing from the model? If yes, please state the subdimension with a brief description.

2. In your opinion, is there a **quality aspect** (unbordered text) missing from the model? If yes, please also state in which subdimension you would position it.

3. Looking at the whole model, are there any existing quality aspects that you would like to position in a different subdimension than the one they are in right now? If yes, please state the quality aspect and the more fitting subdimension.

Lastly, they were given the chance to leave comments of any kind before they were reminded of the data protection details and the study was concluded.

5.2 Participant Acquisition and Demographics

The participants were acquired through a mixture of convenience sampling in a personal and academic environment, as well as using industry contacts to advertise the survey to software teams in the software departments of two different German companies. The survey was exclusively advertised to candidates who had at least some experience developing software in teams. Students without external work experience were only allowed participation if they had at least completed the fifth semester of the undergraduate program, including the completion of the *Software Projekt*, a semester long project where students worked in teams to develop software for real world customers. To aid the acquisition, the potential participants were monetarily incentivized in the form of a gift card raffle for completing the survey. The survey was conducted in English, using the online survey tool *LimeSurvey*[1]. The survey used cookies to remember participants in case they paused and later continued the survey and to avoid duplicate participation. The response sets were saved anonymously, but coherently for multiple reasons:

[1] https://www.limesurvey.org/

First, the monetary incentive might motivate thoughtlessly rushing through the answers, yielding low quality results. Therefore, a response set needed to be relatable to the completion time to filter out those candidates. Since the survey was planned to take around 15 to 20 minutes, any response set completed in under four minutes was deemed invalid. Secondly, since a comparison of answers between lower and higher experience software developers was planned, the responses also needed to be relatable to the participant's work experience.

A total of **117 people** clicked on the survey link, though many closed the survey again before completing the CAPTCHA and at least some of the clicks were duplicates from the same individuals. Since cookies were activated, it is not quite clear how this happened and might have either been a technical limitation of the tool or due to some of the participants using anti-tracker tools in their browser. During the first section of the survey (personal questions and explanation), **31 unique participants aborted** the survey. This confirmed the previously discussed concern, which was heavily considered in the design of the study, despite best efforts being made not to lose too many people during the explanation. Only two further people did not finish the study after getting through the explanation. A total of 61 unique participants completed the survey, three of which completed the survey in (significantly) less than four minutes, deeming their response sets invalid. This left a total of **58 valid response sets**.

The average age of the participants was **27.4 years**, with the oldest being 60 years old and the youngest being 20. The average working experience with software development was **4.7 years**, the highest being 31 years and the lowest being 0 years (students, who completed the *Software Projekt*, but did not have any external work experience, were asked to state their work experience as 0 years). Further demographic details are shown in figure 5.2. Unsurprisingly, a correlation between age and work experience can be seen, with the majority of participants being under the age of 29, with 3 years or less in work experience. However, as figure 5.2 shows, there was at least one participant representing roughly every age and experience bracket within the population, up to the oldest participant with an age of 60 (which is just a few years under retirement age in Germany). The study population was divided into two groups, based on work experience, to allow for comparison and to avoid the majority participant demographic opinion to outweigh the opinion of the more diverse minority demographic. Otherwise, if all ratings were only evaluated as a composite average of the entire study population, the findings would likely favor the priorities of the less experienced, younger developers. Therefore within the following evaluation, the population was divided as evenly as possible and analyzed separately in addition to the composite results. The division was based

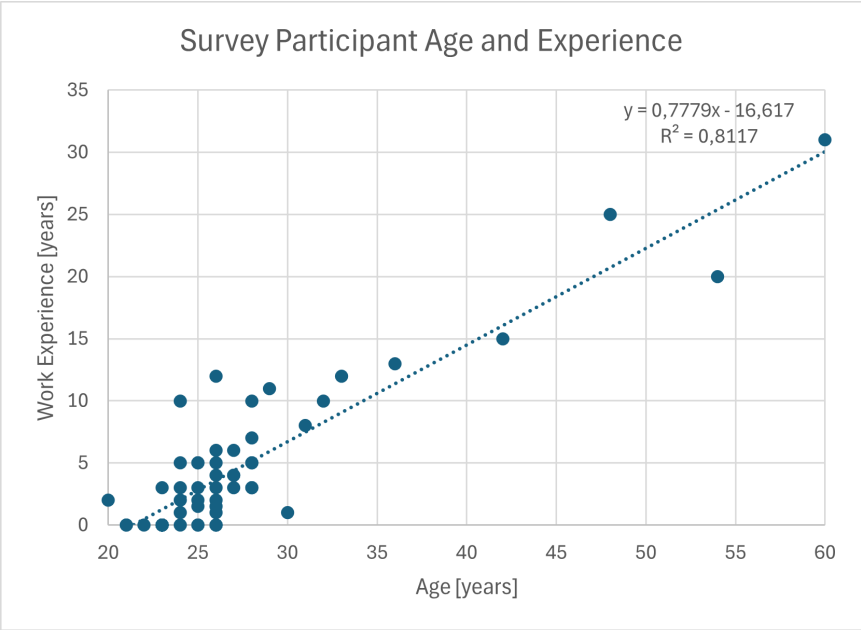


Figure 5.2: Survey participants' age and corresponding work experience.

around the median work experience of 3 years (which also roughly subdivides the group based on age by correlation): Participants with less than three years of experience were placed in the first group (LowExp), representing experienced software development students and professional newcomers. Participants with three or more years of experience were placed in the second group (HighExp), representing experienced professionals. The resulting LowExp group consisted of 27 participants with an average experience 0.51 years, the HighExp group of 31 participants with an average experience of 8.35 years.

5.3 Results

This section serves the purpose of presenting the data gathered from the survey study and commenting on noteworthy findings. The implications of all finding, including the free-text answers given during the survey, will later be discussed holistically in chapter 6.

The average quality aspect rating given in the survey was **7.49**, with a standard deviation of **1.87**. The average rating of the LowExp group was 7.71, the average rating of the HighExp group was 7.31. The average difference in rating between LowExp and HighExp was 0.48 points per quality aspect. The overall average ratings for all quality aspects are visualized in figure 5.3, ranked highest to lowest.

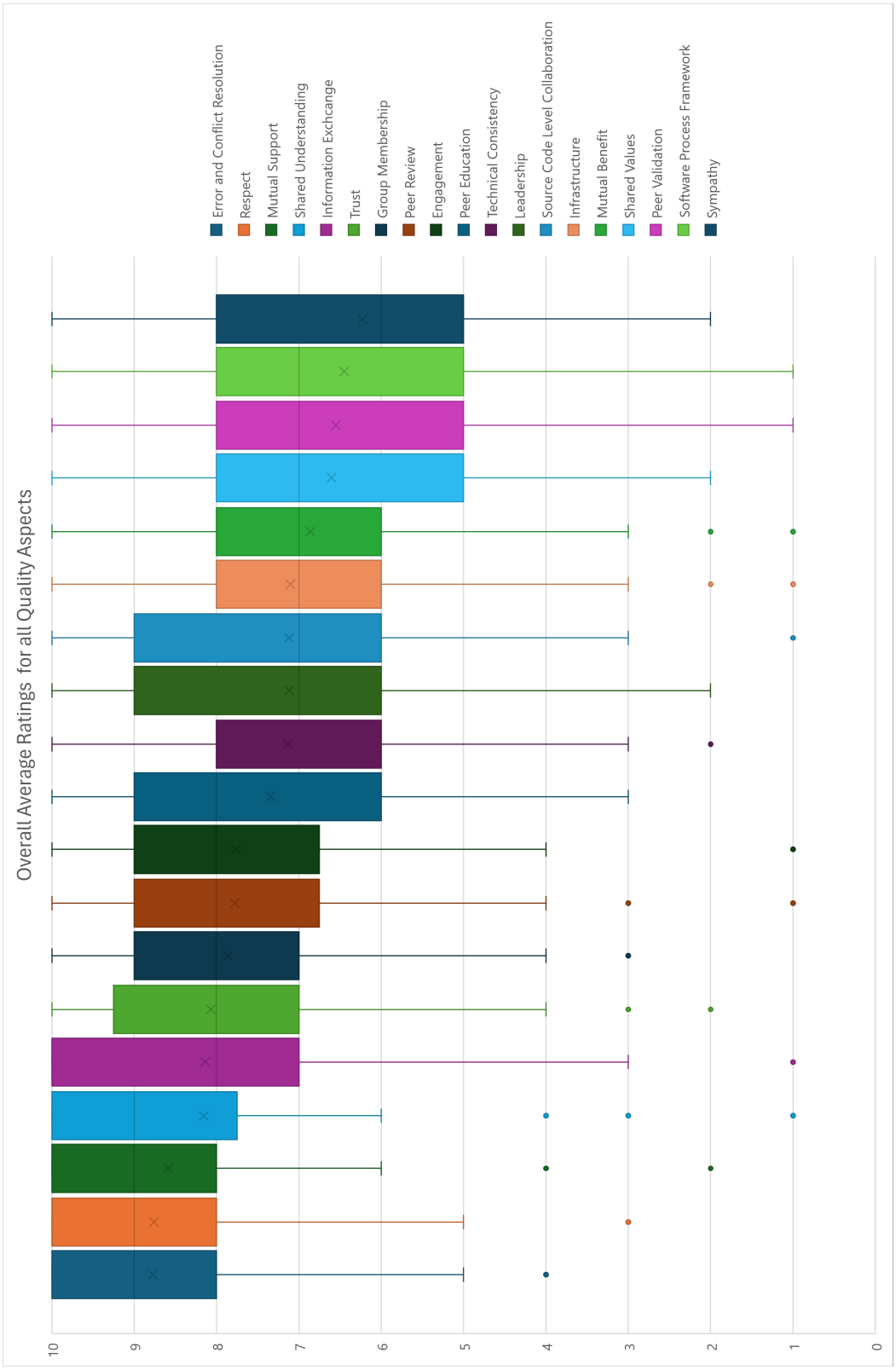


Figure 5.3: Ratings of quality aspects, sorted by highest rated to lowest rated aspect.

5.3.1 Coordination

The average rating for quality aspects of *Coordination* was **6.89**. The average ratings, from high to low, were as follows: *Leadership* (7.12), *Infrastructure* (7.10), *Software Process Framework* (6.45). The quality aspects of *Coordination* were among the more controversial, with *Software Process Framework* and *Leadership* having standard deviations of 2.26 and 2.15 points respectively.

The ratings of the LowExp and HighExp groups are visualized in figure 5.4. The participant groups were relatively similar in their ratings. The biggest divergence of the group opinions was *Infrastructure*, with the LowExp group rating it an average of 0.78 points higher than the HighExp group. *Software Process Framework* and *Leadership* were two of the few quality aspects the HighExp group rated higher than the LowExp group.

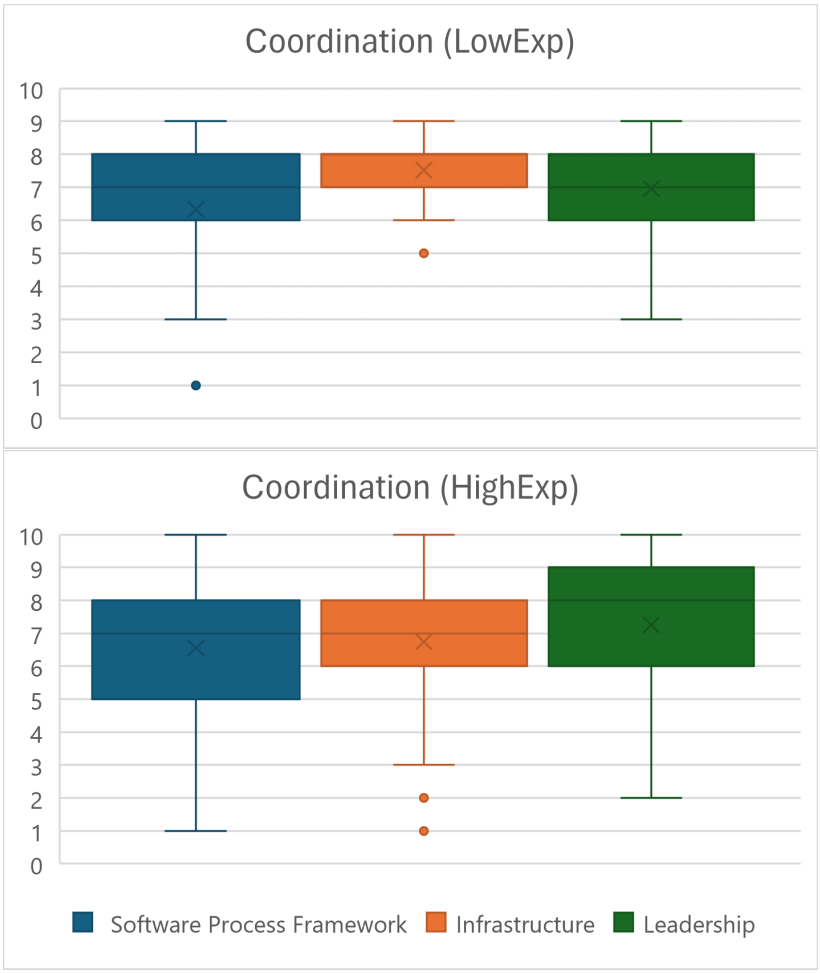


Figure 5.4: Ratings for *Coordination* quality aspects.

5.3.2 Peerage

The average rating for quality aspects of *Peerage* was **7.56**. The average ratings, from high to low, were as follows: *Mutual Support* (8.59), *Peer Review* (7.78), *Peer Education* (7.34), *Peer Validation* (6.56). The quality aspects were relatively uncontroversial.

The ratings of the LowExp and HighExp groups are visualized in figure 5.5. The differences for *Peer Education* and *Peer Review* were very small between groups. Bigger differences can be seen in *Peer Validation* and *Mutual Support*, with the HighExp group rating both an average of 0.91 points lower.

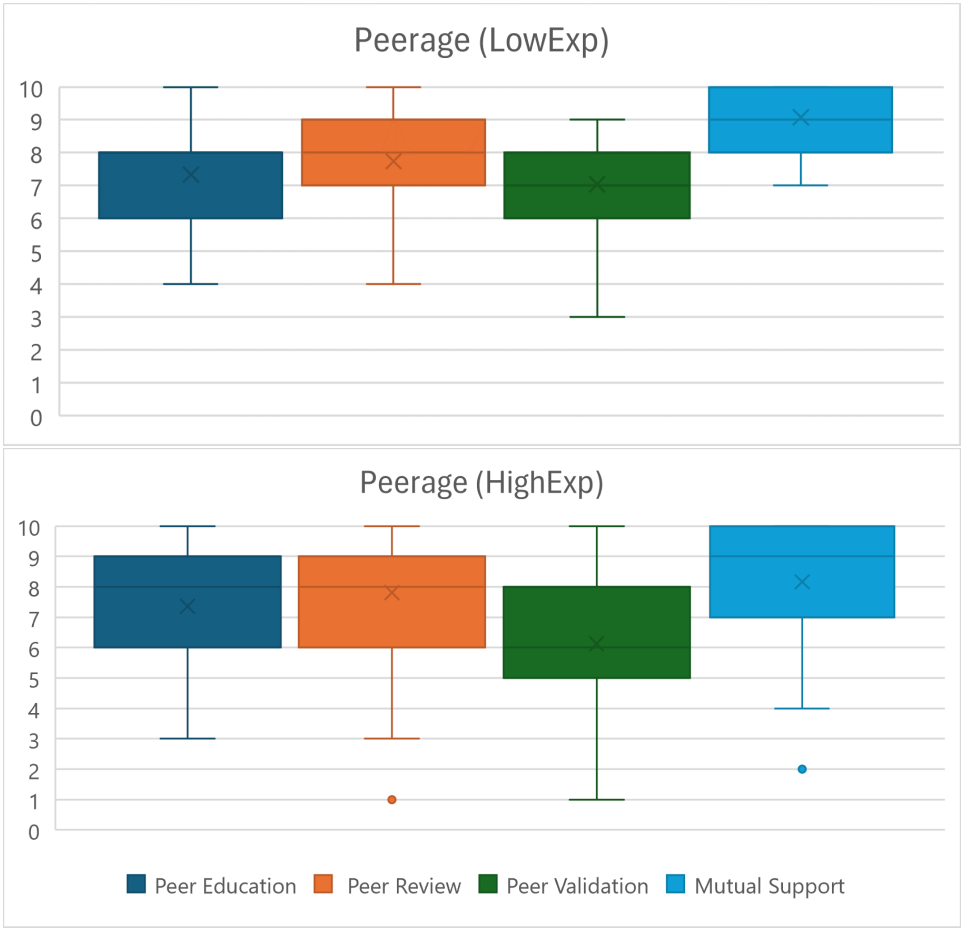


Figure 5.5: Ratings for *Peerage* quality aspects.

5.3.3 Social Coherence

The average rating for quality aspects of *Social Coherence* was **7.72**. The average ratings, from high to low, were as follows: *Respect* (8.76), *Trust* (8.07), *Group Membership* (7.86), *Sympathy* (6.22). The quality aspects were uncontroversial, however *Sympathy* was rated significantly lower than the rest, being the lowest rated quality aspect of the entire model.

The ratings of the LowExp and HighExp groups are visualized in figure 5.6. The differences for ratings were relatively small between groups. Merely *Trust* was rated noticeably higher by the LowExp group, with an average of 0.78 more points. *Group Membership* was much more controversial for the HighExp group.

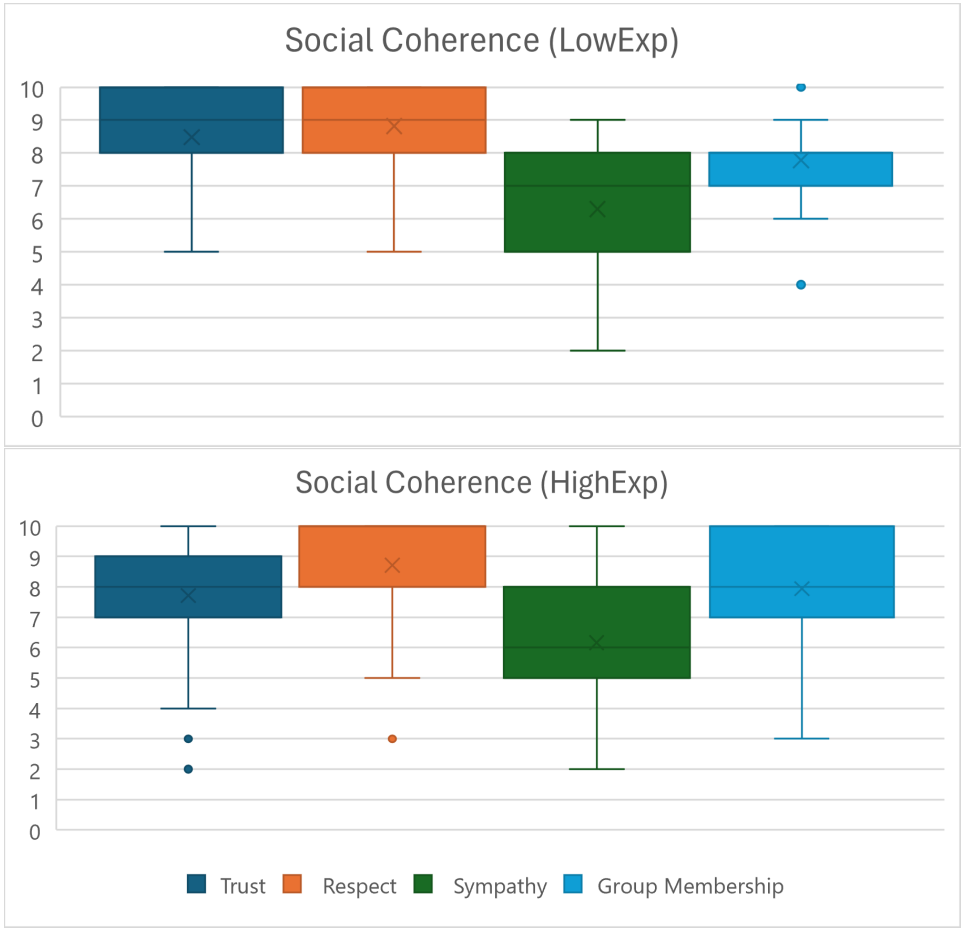


Figure 5.6: Ratings for *Social Coherence* quality aspects.

5.3.4 Communication

The average rating for quality aspects of *Communication* was **8.36**, making it the highest rated overall subdimension. The average ratings, from high to low, were as follows: *Error and Conflict Resolution* (8.78), *Shared Understanding* (8.16), *Information Exchange* (8.14). The quality aspects were uncontroversial, with *Error and Conflict Resolution* being the highest rated quality aspect of the entire model, while also being the least controversial with a standard deviation of only 1.39.

The ratings of the LowExp and HighExp groups are visualized in figure 5.7. The differences for ratings was relatively small between groups for all quality aspects, although *Information Exchange* was considerably more controversial within the HighExp group.

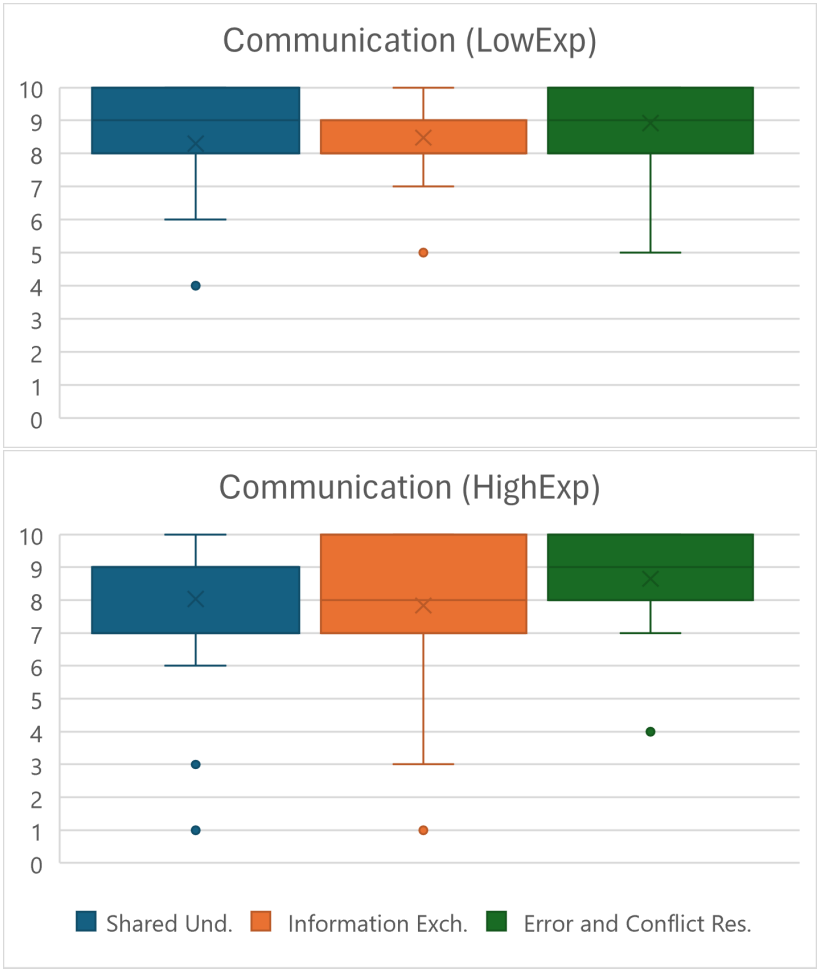


Figure 5.7: Ratings for *Communication* quality aspects.

5.3.5 Motivation

The average rating for quality aspects of *Motivation* was **7.07**. The average ratings, from high to low, were as follows: *Engagement* (7.76), *Mutual Benefit* (6.86), *Shared Values* (6.60). Overall, the quality aspects were relatively uncontroversial in the LowExp group but controversial within HighExp group.

The ratings of the LowExp and HighExp groups are visualized in figure 5.8. The HighExp group rated the quality aspects for *Motivation* an average of 0.88 points lower than the LowExp group. This makes the importance of *Motivation* the biggest disagreement between the groups, with *Shared Values* being the most disagreed upon quality aspect, being rated an average of 1.23 points lower by the HighExp group.

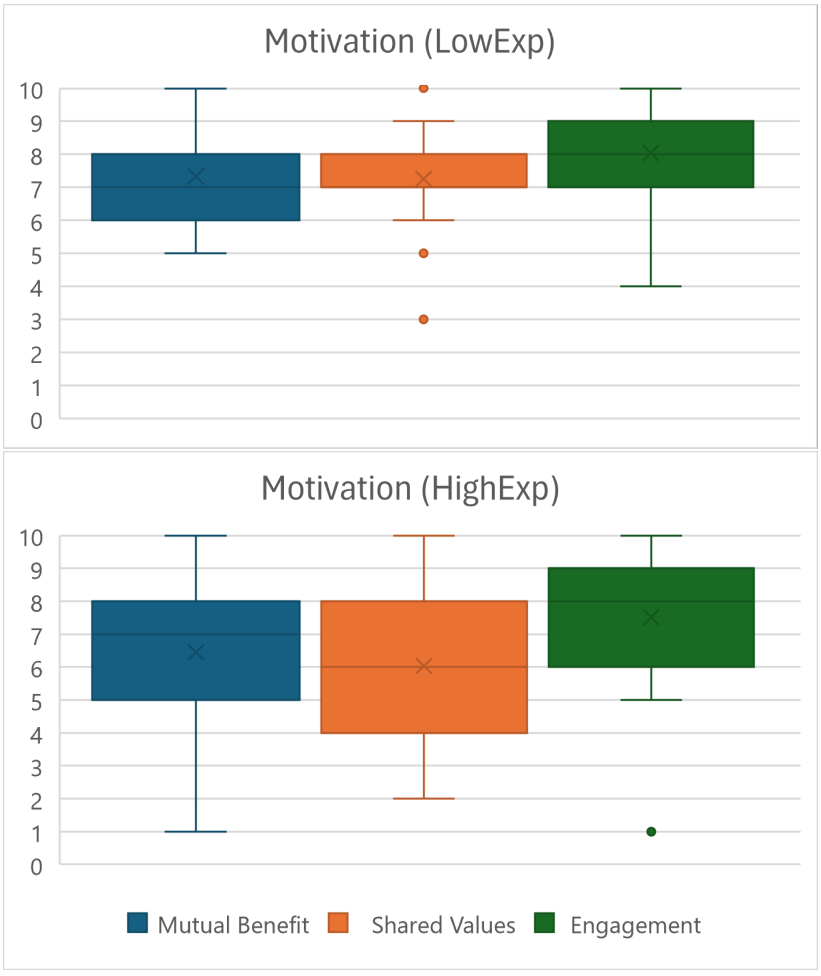
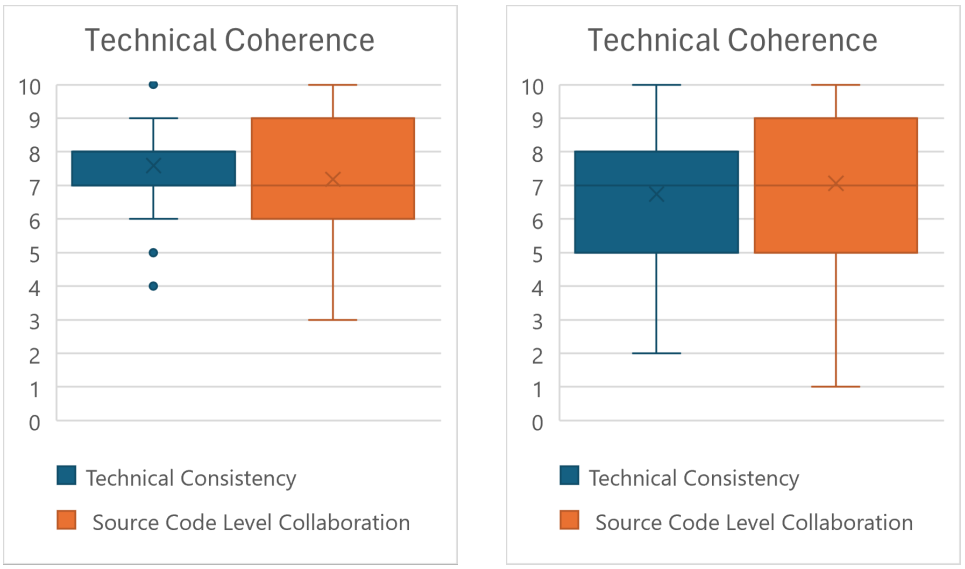


Figure 5.8: Ratings for *Motivation* quality aspects.

5.3.6 Technical Coherence

The average rating for quality aspects of *Technical Coherence* was **7.13**. The average ratings, from high to low, were as follows: *Technical Consistency* (7.14), *Source Code Level Collaboration* (7.12). Both quality aspects were among the more controversial, with *Technical Consistency* and *Source Code Level Collaboration* having standard deviations of 2.06 and 2.21 points respectively.

The ratings of the LowExp and HighExp groups are visualized in figure 5.9. The participant groups rated *Source Code Level Collaboration* very similarly with a difference of 0.12 between groups, while they valued *Technical Consistency* relatively differently, with the HighExp group rating it an average of 0.85 lower.



(a) LowExp Group (b) HighExp Group

Figure 5.9: Ratings for *Technical Coherence* quality aspects.

5.3.7 Intra- and Inter-Participant Variance

The results showed higher variance in the ratings within the HighExp group compared to the LowExp group, with the ratings of the HighExp group having a standard deviation of 2.09, while the ratings of the LowExp group only had a standard deviation of 1.55. This initially appears to strengthen the previous hypothesis that more experienced programmers are more opinionated, having stronger and more clearly defined opinions on what is important for collaboration due to their work experience. However, when looking closer at the average intra-participant variance (the standard deviation of the ratings of a single participant), the difference between groups is much smaller, with the HighExp and LowExp group having an average intra-participant variance of 1.77 and 1.57 respectively. Conclusively, the increased variance in the ratings of the HighExp group is likely just due to individual outliers who rated most aspects significantly lower than the rest of the group. Therefore, the data does not indicate a difference in strength of opinion between the two groups based on their ratings.

Chapter 6

Discussion

The interview and survey study showed, that different developers have strong opinions regarding different quality aspects of the model. This chapter will discuss all previously gathered data, including the yet to be discussed free-text answers of the online survey, which led to many valuable insights. A total of 45 free-text answers were submitted in the survey. Subsequently, final conclusions about the model will be presented, including facets that could be considered for future iterations of the model, as well as exclusions from the existing model.

6.1 Importance of the Individual Quality Aspects and Subdimensions

As already visually presented in figure 5.3, the survey yielded a clear ranking of the quality aspects, presented again with their average ratings in table 6.1 for closer inspection and discussion. While the HighExp group ranked most

#	Quality Aspect	Rating	#	Quality Aspect	Rating
1	*Err. and Conflict Res.*	8.78	11	*Technical Consistency*	7.14
2	*Respect*	8.76	12	*Leadership*	7.12
3	*Mutual Support*	8.59	13	*SC Level Collaboration*	7.12
4	*Shared Understanding*	8.16	14	*Infrastructure*	7.10
5	*Information Exchange*	8.14	15	*Mutual Benefit*	6.86
6	*Trust*	8.01	16	*Shared Values*	6.60
7	*Group Membership*	7.86	17	*Peer Validation*	6.55
8	*Peer Review*	7.78	18	*SP Framework*	6.44
9	*Engagement*	7.76	19	*Sympathy*	6.22
10	*Peer Education*	7.34			

Table 6.1: Rankings of quality aspects with their respective average ratings.

quality aspects lower than the LowExp group on average, the overall rankings of quality aspects were very similar between the two groups and will therefore not be discussed separately from here onward. None of the quality aspects were considered entirely unimportant, indicating a good selection of quality aspects. While no quality aspect was averagely ranked lower than *6*, there was a clear difference between the quality aspects considered most and least important, with *Error and Conflict Resolution* (8.78) and *Respect* (8.76) being considered the most important, while *Sympathy* (6.22) and *Software Process Framework* (6.44) were considered an average of around 2.5 points less important. Generally, the participants ranked social and interpersonal factors as more important for collaboration than technical factors, with one HighExp participant explicitly stating this in a comment:

"In my experience the human and social components of the software project will have the most impact [...]."

This is also reflected in the overall rankings of subdimensions presented in table 6.2 (the ratings of the subdimensions are derived from the composite averages of all its quality aspects). *Communication* and *Social Coherence*

#	**Subdimension**	**Rating**
1	*Communication*	8.36
2	*Social Coherence*	7.72
3	*Peerage*	7.56
4	*Technical Coherence*	7.13
5	*Motivation*	7.07
6	*Coordination*	6.89

Table 6.2: Rankings of subdimensions with their respective average ratings.

were both classified as purely social subdimensions and considered the most important, with *Peerage* being a close third as a socio-technical aspect. While *Motivation* was also classified as a social aspect, it was not considered as important as the others. This could be due to *Motivation* being more of a personal factor, while the participants rated the quality aspects most highly that revolved around the direct social interaction of team members. However, this does not mean that the more technical aspects are all negligible, with many participants rating them with *10* as well. Some participants were even outspoken about their importance in the free-text answers, with one HighExp participant commenting on *Technical Consistency*:

"Only equal technical requirements enable, that the team can collaborate. Shared approaches/shared problems."
(Translated from German)

However, based on the ratings and some further free-text answers, certain lower rated quality aspects needed to be re-evaluated.

6.2 Low-Rated and Critiqued Quality Aspects

Of all quality aspects, four aspects were ranked considerably lower than the rest, with a minimum 0.26 point difference to the next highest quality aspect: *Shared Values*, *Peer Validation*, *Software Process Framework* and *Sympathy*. Moreover, three of these quality aspects were also all criticized explicitly, which was generally a rare occurrence in the validation process and raises the need for discussion.

Shared Values was rated fourth lowest, with an average rating of 6.60. One participant even suggested the exclusion of the quality aspect:

"The aspect "Shared Values" can be excluded from the model entirely, as the premise behind the aspect is not logical for all cases. For example, despite two developers having a different value system and working style in relation to their work, the differences might even unknowingly become a good fit for the project and might be mutually beneficial as each developer would solve a different problem/aspect within the project, instead of conflicting with each other due to their differences."

This is a valid point, although the participant refers to the diversity of values inside a team as valuable, suggesting that the developers should work on different problems within the project if they have conflicting values. This is something that concerns good teamwork and team compositions, but is counterproductive for collaboration, as differentiated in chapter 2.2. However, it is true that it may not be ideal for a project if all developers have the same technical priorities and is something that needs to be considered. Another possible reason for the lower ratings is, that the explanation for each quality aspect had to be kept short. For *Shared Values*, an example was included in the explanation that described a difference in technical values between two developers. However, as explained in chapter 3.6.2, the aspect also describes shared moral values and ideals. These might have been overshadowed in the survey explanation and might otherwise have been rated higher. One participant even commented on the distinction:

"Maybe separating moral and technical shared values in different aspects could be useful; the technical part could then also fall into one of the technical categories."

This is a very good suggestion, the resulting revision of which will be discussed in detail in the following chapter.

Peer Validation was ranked relatively low with an average of 6.55, however no participant commented negatively on the aspect. In fact, some participants explicitly suggested adding "appreciation" as an aspect to the model, which is a facet of *Peer Validation*. This is also congruent to statements from the interview study and perhaps, again, due to having to keep the explanations in the survey very brief. Since there was no explicit criticism and since there is still adequate reason to believe that validation by peers is something that people value in software teams, the aspect was not excluded from the model based on ratings alone.

Software Process Framework was ranked second lowest with an average rating of 6.44 and was also explicitly criticized. Two HighExp participants stated the following:

"My perspective might be biased as I usually consult in larger organizations and frequently the only way to get things done effectively is working around their processes while making it seem like you're following them."

"A lot of software processes and infrastructure topics cost a lot of time which is better used for the development which will greatly improve the quality of the software."

These opinions could come from a general distaste for too high of a focus on frameworks in organizational contexts, however they raise questions regarding the justification for the presence of the aspect. The initial motivation for including *Software Process Framework*, was that the framework sets the ground rules and constraints for multiple collaboration-relevant processes, such as communication (through meeting structure), collaboration practices (such as code reviews and pair programming) and some coordinating elements like progress reporting and iteration frequency. However, the first two are already adequately described after revisions to the model, with the collaboration practices all being included in the *Peerage* subdimension and describing them again through the aspect would be redundant. The progress reporting and iteration frequency elements were mostly included, because they were shown to increase motivation, or to be more exact, the engagement of the developers. Consequently, those two facets should just be included as contributing factors for *Engagement*, while all other facets of *Software Process Framework* are already adequately described by the model. Therefore ***Software Process Framework*** **was excluded from the model** based on survey ratings and feedback.

Sympathy was clearly the lowest rated aspect of the entire model with an average rating of 6.22. It was not commented on at all in the interview study and also explicitly criticized in the survey. One HighExp participant explained the following:

"The aspect "Sympathy" is a morally noble quality and its inclusion in this collaboration quality model is well-appreciated, although it might not be appropriate to consider this aspect for professional scenarios. Software engineering and development is a demanding profession which requires a skilled workforce to undertake complex yet ambitious real-world projects and implement efficient solutions in time. A trait like "Sympathy", on the contrary, might even be unprofessional to harbor to a great level, as it might unwillingly induce complacency in teams and help unidentified errors persist in projects."

The participant explains how sympathy might be a good concept in theory, but can lead to many problems in professional scenarios, as it might induce complacency or lead to unidentified errors persisting in projects. Since the purpose of the model is to eventually be used to improve collaboration in real software teams, this is very valuable feedback. *Sympathy* was already one of the weaker aspects included in the model, since it is hard to measure and influence, therefore not being the best fit for inclusion in a quality model. Also, while high sympathy within a team can be both good and bad, the collaboration should be able to function regardless of the interpersonal liking of the team members. Consequently, prompted by the feedback and the fact that it was the lowest rated quality aspect, ***Sympathy* was excluded from the model.** However, the suggested measurement scale for *Group Membership* will include *Sympathy* as an item (cf. chapter 7), as it is not entirely negligible for the dynamic of the group.

6.3 Considerations for Revisions

As previously discussed, one participant raised the point, that *Shared Values* should be divided into shared technical values and shared moral values. Indeed, the two facets are different enough to be worthy of revising the aspect. As another participant noted, in the case of technical values it may not be ideal that they are all identical within a team, rather that they are congruent with one another. The technical values are something that impacts the technical cohesion, such as when two developers work on the same software module and have conflicting values in regards to technical priorities (i.e. code readability versus efficiency). In this case, it is most important that the team jointly decides which values are important for the project (or the individual software module) and then to apply them *consistently*. Therefore, the technical value congruence should be included in the aspect *Technical Consistency*. Meanwhile, the shared values that lead to an actual increase in motivation are shared ideals of the team members in regards to the collaboration, the team and the project. Are the team members satisfied with the direction of the project? Are the team members happy with how the team is being led and organized? Are the

team members on the same page when it comes to frequency and degree of collaboration within the team? Do the team members have a shared mission or goal? Therefore, ***Shared Values*** **will be revised into** ***Shared Ideals*** **in the model.**

Two facets which have been repeatedly mentioned in both interview and survey study, which are not directly depicted in the model, were "openness" (or "transparency") and "appreciation". These facets were not ascertained by the literature analysis. However, with the same reasoning as in the interviews, the facets will not be included explicitly as quality aspects. Openness is likely the direct result of good *Social Cohesion* (*Trust*, *Respect* and *Group Membership*) and a good *Error and Conflict Resolution* culture. Including it again separately would be redundant. Appreciation by peers is included in *Peer Validation*, while appreciation from an organizational perspective is included in *Leadership* (application of organizational politics). The effects of appreciation are most relevant for the engagement of a developer, which is a quality aspect already included in the model as well.

Furthermore, some comments were made that suggested adding aspects to the model which were already clearly intended to be included (such as one participant suggesting adding "Organizational Policies", which has been defined as a clear component of *Leadership*). This is likely, again, due to the format of the online survey requiring brief descriptions, which were not able to convey the full definitions of each aspect without losing the majority of potential participants. However, they further strengthen the existing quality aspects they refered to. Another participant suggested adding "Cultural Aspects" as a subdimension. While it has already been discussed that sociocultural distance of the team has an impact on collaboration, the suggestion will not be included since the model follows the approach of being applicable to all types of teams, both culturally diverse and homogeneous.

6.4 Interdependence of Socio-technical Factors

As already discussed in the fundamentals (cf. chapter 2), the interdependencies between influencing factors in socio-technical systems present a challenge for the creation of a quality model, since it requires some degree of categorization. The survey study further cemented this fact through the statements of various participants. While most participants agreed with the placements of the quality aspects in the subdimensions, many left comments nevertheless, stating that a quality aspect would *also* fit into a different subdimension:

"I think that shared understanding is correctly positioned here, but might also be concerned with social cohesion."

"Mutual support seems also to fit in social cohesion or communication I think"

"I think trust and respect can also be parts of communication, but I think they rather fit here."

"Source Code Level Collaboration could also be a part of coordination (because of exemplary interactions through git etc.)"

"Depending on the project, the leadership might have a huge influence on the shared values and overall motivation."

"I think for Peerage, it is important for peers to celebrate their achievements as a group. Maybe that falls in Group Membership"

These comments do not necessarily provide any specific new insights, but demonstrate how software developers link together the quality aspects and subdimensions based on their experience, which solidifies them as important parts contributing to good collaboration in the real world. Some participants also very adequately described the interdependencies explicitly:

"Some quality aspects could be part of more than one larger subdimension. For example: Group Membership contributes to Motivation. If you feel like you don't belong to the group and are left out your motivation to participate in the project goes down. Or peer education contributes to social cohesion because it mostly improves respect, trust and sympathy for one another."

"The quality aspects are also dependent among one another. There are hygiene-factors, which have to be satisfied, like for example a certain level of salary, so that other aspects become relevant, i.e. appreciation of one's work. [...]"
(Translated from German)

"These categories are usually very close coupled [...]."

"I think that a lot of these quality aspects are hard to be evaluated alone. In large software projects these aspects will almost always influence each other. Its worth to analyze these influences which would allow more insight into the relation between quality and these categories. It would be also worth to explore the factors of lower quality and to match these with the categories or to derive additional categories."

Especially the last comment, while also describing the interdependencies, exemplifies the need for quality models describing socio-technical aspects such as collaboration. Due to the strong influences between the aspects, deficiencies in socio-technical factors can rapidly lead to downward spirals in software teams. Therefore, models need to exist to analyze the constituents, so that the problems can be more accurately identified, instead of just being able to observe the result of good or bad collaboration as a whole. Furthermore, the participant also raises a good point about possible future research for the model: Identifying the exact influences between the quality aspects, including the degree of influence or the impact of patterns that lead to low-quality collaboration, could lead to even more insight into collaboration and help more accurately identify and prevent problems in real software teams.

6.5 Positioning Suggestions and other Comments

Many comments were left regarding the positioning of the quality aspects within both the subdimensions and the planes of the model (technical, socio-technical and social). However, the only suggestion that was left by more than one single participant was moving *Peer Education* from the technical to the socio-technical plane.

"If peer education describes not only the technical aspect of organization and documentation of useful information, then I would move it to the socio-technical dimension. I think it also implies the willingness of team members to educate their peers, which would include some level of sympathy or helpfulness."

This is well argued and since a lot of education between peers happens through interpersonal interactions, the suggestion was accepted for the model.

One participant suggested adding additional colors, for the purposes of sub-categorization, to make the model less overwhelming on first sight. However, other comments explicitly praised the model for being clear and easy to understand (though there may be some degree of survivorship bias, since all comments came from people who stuck with the survey to the end). Due to the natural complexity of quality models, some people finding the model overwhelming at first may be a fact that has to be accepted.

6.6 Summary of Revisions and Final Model

Based on all feedback gathered, the following changes were made for the final draft of the model:

- *Sympathy* was excluded based on low ratings and criticism.
- *Software Process Framework* was excluded based on low ratings, redundancy and criticism.
- Progress reporting and iteration frequency, previously facets of *Software Process Framework*, were noted as contributing factors to *Engagement*.
- *Shared Values* was changed into *Shared Ideals* based on feedback, with technical value congruence being noted as a facet of *Technical Consistency*.
- *Peer Education* was moved from the technical plane into the socio-technical plane.

All changes are included and visualized in figure 6.1.

6.7 Threats to Validity

Throughout this thesis, most threats to validity have already been discussed in their respective chapters. The first synthesis of the model, based on literature, had some degree of arbitration. However, this was necessary to create a starting point for later validation and discussion of a quality model. The interview study had a relatively low participant number, but due to the high diversity of work experience in the participants they were deemed sufficient to validate the model for completeness. The survey study had an adequate participant number, but a bias towards younger, less experienced developers, which could have led to a bias towards their opinion in composite ratings. This was counteracted by dividing the participant group and analyzing the results separately. Furthermore, despite the bias, there was at least one representative for each age and experience bracket. Lastly, the final revisions to the model, while basing them on holistic analysis of all gathered information, have not been validated yet. While threats to validity were considered and handled in best conscience, even richer data and validation could be gathered in future work, which will be discussed in chapter 8.

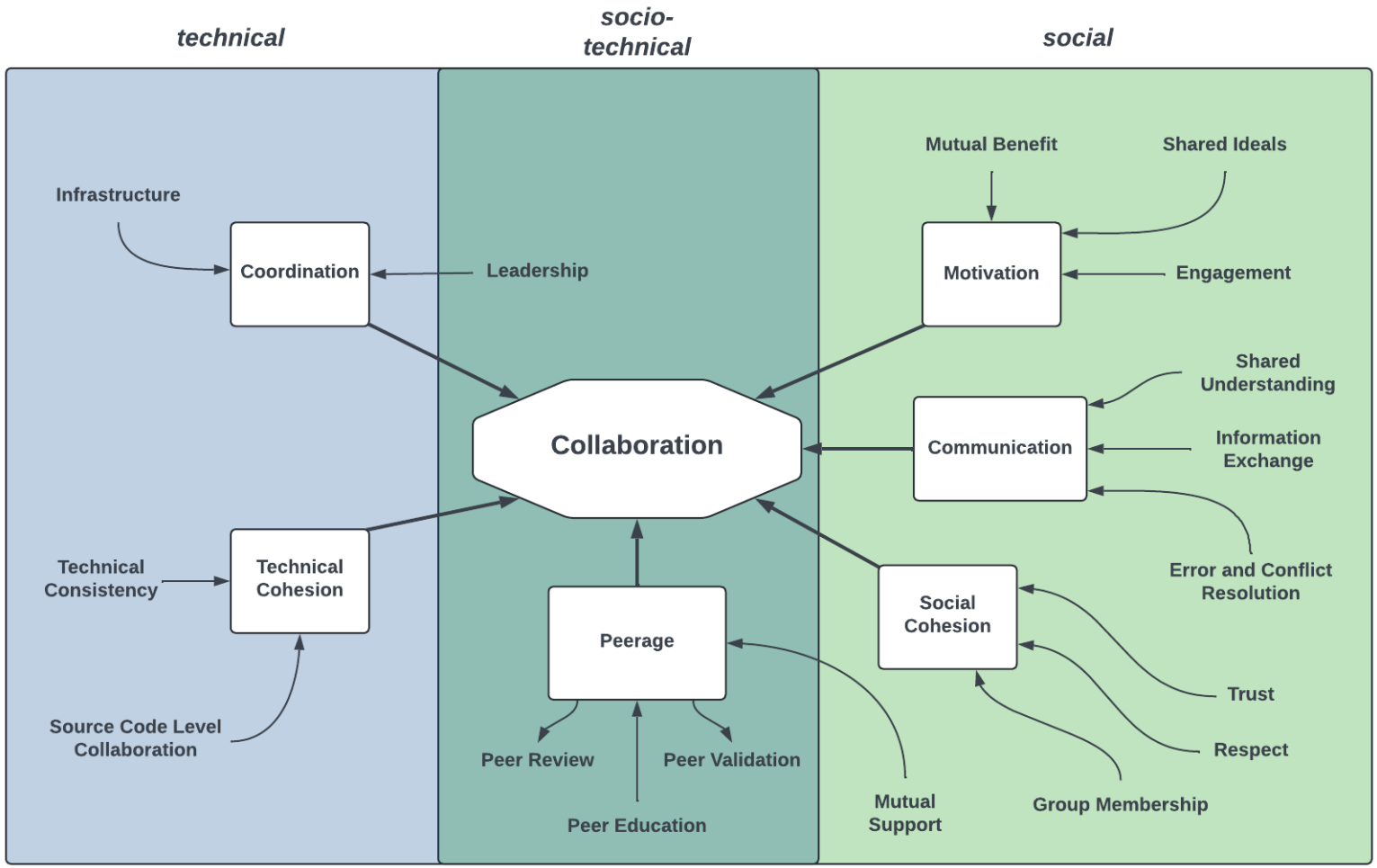


Figure 6.1: Final visualization of the quality model for collaboration, based on holistic discussion of literature, interviews and survey feedback.

Chapter 7

Collaboration Metrics

As explained in chapter 2.1, quality models usually do not include specific metrics for their quality aspects. Instead, application of the model includes determining which specific facets to measure by which means through processes like GQM. Nevertheless, this chapter will serve to supplement the developed quality model with metric suggestions for each quality aspect included in the final iteration of the model (cf. figure 6.1). This is for the purpose of giving potential practical applicants a first starting point for the usage of the model, as well as being a proof of concept that the included quality aspects are indeed measurable. Ideally, objective metrics established in literature were desired for all quality aspects. However, due to the socio-technical nature of most aspects, most metrics are based on survey scales. Since this model tries to cater to the specific needs of software teams, many established survey scales were considered sub-optimal for the application in software teams. For aspects, for which no satisfying established survey scales could be found in literature, a new survey scale was constructed, with the items being inspired by data gathered in the interview and survey. All suggested questionnaire items are intended as parts of Likert-type scales (statements for which the degree of agreement is to be expressed).

7.1 Infrastructure

Infrastructure was an aspect described in literature, but not attempted to be measured. The quality of the infrastructure corresponds to the degree in which the infrastructure accommodates the needs of the developers. Following questionnaire items are suggested, based on categorization by Whitehead et al. [29] and developer feedback:

1. The available tools allow me to communicate with my team effectively.
2. The available tools support the technical collaboration of me and my team members.

3. I am informed about the work status of my team members and the project as a whole.

4. The available tools help me and my team to coordinate our work.

5. My work environment (physical or digital) is conducive to collaboration.

7.2 Leadership

Team leadership is a concept attempted to be measured by literature, however it is often highly specified to the leadership style. Therefore, instead of suggesting a metric for a particular leadership style, own questionnaire items are suggested based on what developers noted as important in regards to leadership in interview and survey comments.

1. The leadership supports the team in cases of unresolved conflicts or disagreement.

2. The leadership tries to encourage and support collaboration between me and my team members.

3. The leadership offers direction for the project, without restricting my team too much in its freedoms.

4. My team is considered an important part of the organization and is appreciated accordingly.

7.3 Technical Consistency

Since this aspect was derived from the interview feedback, no suitable metric could be found in literature. The following items are suggested based on developer opinion:

1. Whenever possible, all members of the team use the same software tools and frameworks for the same needs.

2. My team has clearly established coding conventions which are consistently enforced in the project.

3. My team agrees on which technical values take priority in which parts of the project.

4. The technical expertise of my team members is adequate for the discussion of necessary technical topics.

7.4 Source Code Level Collaboration

As presented by Caglayan et al. [6] (cf. chapter 3.5.1), *Source Code Level Collaboration* can be measured through collaboration networks, which can be automatically extracted by VCS software. The metrics used correspond to the metrics of the graph itself, for example degree, centrality, or betweenness of the individual developers. These metrics can be used, to identify collaboration patterns and weaknesses, such as bottlenecks or vital collaboration bridges.

While some literature describes a fully connected collaboration network as the goal, representing collective code ownership, each team has to determine their own goals, based on the difference of specialization within a team. The network metrics can be used to objectively quantify dynamics of the underlying technical collaboration within a team and are therefore ideal as metrics for this quality aspect.

7.5 Peer Review

Peer Review is represented through the practices of code reviews and pair programming in software teams. The occurrence of code reviews is something that can (and should) first be measured objectively, through ascertaining how much of the code in the team repository underwent review processes before being approved. There are also further objective metrics which can be gathered, such as the average review approval rate, or the average amount and length of review comments.

However, even in teams where code review is an established practice, the quality of the review process can differ based on more subjective metrics. Therefore, the following questionnaire items are suggested to ascertain the quality of *Peer Review*:

1. Nobody in the team is above the code review process.
2. I review my peers' code carefully and to the best of my conscience.
3. I have the impression that my own code is reviewed carefully.
4. Comments made by my peers are sensible and meaningful.
5. Reviewing the code of my peers can be an insightful and educating experience.

7.6 Peer Education

Peer Education was a quality aspect inspired by anecdotes about industry practices and has therefore no established metric found in literature. Following questionnaire items are suggested:

1. I frequently learn new things from my peers.
2. We have an adequate amount of events and practices, which are centered around the shared or reciprocal education of me and my team members.
3. If I wish to learn more about a part of the project, there exists adequate high level documentation or equivalent resources for me to do so.

7.7 Peer Validation

Peer Validation was described as a special dynamic in software teams by Augustin et al. [1], however no established metrics could be found to measure it. The following items are suggested based on the findings of Augustin et al. and developer feedback:

1. I wish to impress my peers rather than my boss with the quality of my work.
2. The quality of my work and my personal abilities are acknowledged and appreciated by my peers.

7.8 Mutual Support

Cheah et al. [8] measured *Mutual Support* as part of their paper *"Mutual Support, Role Breadth Self-Efficacy, and Sustainable Job Performance of Workers in Young Firms"*. They adapted four questionnaire items from previous social interaction research, which are an adequate fit for software teams. These items were adapted, with the addition of a fifth item based on developer opinion. The items are as follows:

1. The team members support and complement each other as well as they can.
2. Discussions among the team members are constructive and beneficial.
3. Proposals and suggestions of team members are respected.
4. I work within a cooperative ambiance.
5. My team members are willing to accept help.

7.9 Trust

Trust in organizational contexts is a concept which underwent a multitude of attempts to measure it in research. McEvily et al. [18] reviewed literature on the topic of measuring trust in organizational research. They found 129 different measures of trust, almost all of which lacked independent replication by a third party. Furthermore, most measures revolved around trust of asymmetric relationships (i.e. manager to employee, customer to vendor). They only identified five noteworthy measures of trust. The authors did not explicitly suggest a metric (neither did any of the papers they identified as noteworthy measure of trust), but conclusively present a framework for measuring trust, shown in figure 7.1. The five measures are highly congruent with the recounting of experiences made by developers within the research of this paper and will therefore be used to construct five questionnaire items:

1. I trust in the competence and responsibility of my team members.
2. I believe that my team members are honest and act in good faith.
3. I trust in the benevolence and integrity of my team members.
4. I trust that my team members will handle their tasks, without the constant need to check up on them.
5. I can rely on my team members.

Authors	*Trustworthiness Beliefs*	*Trusting Intentions*	*Trusting Behaviors*
McAllister (1995)	Competence/ Responsibility Care & Concern		
Cummings & Bromiley (1996)	Good-Faith Effort, Honesty Limited Opportunism		
Mayer & Davis (1999)	Ability, Benevolence, Integrity	Trust	
Currall & Judge (1995)		Communication Informal Agreement Surveillance Task Coordination	
Gillespie (2003)		Reliance Disclosure	

Figure 7.1: Framework for measuring trust by McEvily et al. [18].

Another promising paper, which was released in the same year as the literature analysis by McEvily et al. and could therefore not be contextualized in the validity of trust research, is *"Measuring trust in teams: Development and validation of a multifaceted measure of formative and reflective indicators of team trust"* by Costa et al. [9]. The authors present a well-constructed questionnaire scale for measuring trust in teams, however it consists of 38 items and thus might be too extensive to be used in the scope of this quality model. However, it is still noteworthy for teams who wish to delve deeper into the analysis of trust within the team.

7.10 Respect

While there are some objective indicators of respect, such as fair wages, hierarchical position or equal treatment, it remains an aspect that is hard to measure beyond subjectivity. In their paper *"Unraveling respect in organization studies"*, Grover et al. [11] deconstruct the different constituents of respect in organizations, based on literature research. The most noteworthy distinction is the differentiation between appraisal versus recognition respect. Recognition respect refers to the *"attitude or belief about how people should be treated generally"*. It includes *"being treated politely and not being insulted"*, as well as listening and cooperating as morally equal parties, as they are fundamental matters of human dignity. Appraisal respect, in contrast, refers to respect shown to an individual based on their perceived individual merits. In the context of software teams, both aspects are vital and thus, two items are constructed for both distinctions of respect:

Recognition Respect:

1. I am being treated politely and with respect in my team.
2. My team members listen to me and treat me as an equal.

Appraisal Respect:

1. My individual merits are acknowledged and respected.
2. I am given the impression, that I am an important part of the team.

7.11 Group Membership

Lindsjorn et al. [16] disclosed their measurement items for measuring the group cohesion of software teams. It is deemed adequate to measure *Group Membership* in the scope of the quality model. Items marked with a "*" are weighed negatively:

1. This teamwork is important to the team.
2. It is important to team members to be part of the team.
3. The team does not see anything special in this teamwork*.
4. The team members are strongly attached to the team.
5. All team members are fully integrated in the team.
6. There were many personal conflicts in the team*.
7. There is mutual sympathy between the members of the team.
8. The team sticks together.
9. The members of the team feel proud to be part of the team.
10. Every team member feels responsible for maintaining and protecting the team.

7.12 Shared Understanding

Shared Understanding is an aspect which can be measured by objective numeric metrics through the application of a *Pathfinder Algorithm* on paired comparisons of relevant concepts. A paper outlining this measurement in software teams is *"Measuring Shared Understanding in Software Project Teams using Pathfinder Networks"* by Braunschweig et al. [5]. While the paper describes the process in detail, and should be referred to if a practical application is desired, a short overview will be given: The process starts with a careful concept selection. Explicit terms or phrases used by the team in specification, source code, or interviews, which participate in collaboration in some form are chosen. The terms are then rated on similarity to all other terms by every team member. This rating creates a Pathfinder Network (the nodes representing the concepts and the edges representing their relationship), which can be analyzed mathematically. This way, *Shared Understanding* within a software team can be adequately measured and analyzed.

7.13 Information Exchange

Similar to *Shared Understanding*, *Information Exchange* also has approaches to be analyzed mathematically through graphs. Klünder et al. [14] present a method in their paper *"Modeling and Analyzing Information Flow in Development Teams as a Pipe System."*. In this approach, they model the information exchange in software teams as a pipe system, including both solid

information, which includes long-term, repeatedly accessible information stores such as source code or written documents, as well as fluid information, which includes undocumented meetings, face-to-face communication and implicit knowledge. Their approach allows extracting certain information flow metrics, such as the centrality, degree and betweenness of the developers in the communication network. The model also allows finding critical paths such as bottlenecks, which could lead to loss of information. Due to its complexity, the process and its metrics cannot be fully presented in the scope of this work and the referenced paper should be referred to directly. However, the resulting metrics of the information flow networks by Klünder et al. are considered an excellent fit to measure *Information Exchange* as part of this quality model.

7.14 Error and Conflict Resolution

No established measurements for error and conflict resolution culture could be found in literature. The questionnaire items are based on the accounts of practicing software developers by Sasse et al. [24], as well as the interview and survey studies conducted in the scope of this work:

1. The team does not point fingers when an error was committed.
2. It is more important for the team how a type of error can be avoided in the future, instead of scrutinizing the one responsible.
3. Conflicts in the team are resolved constructively and efficiently.
4. When it comes to conflicts, I do not feel unfairly treated.
5. I feel like I have an equal voice in case of differences in opinion.

7.15 Mutual Benefit

Mutual Benefit revolves around the personal impression of team members, that they gain adequately from the collaboration. Based on some questionnaire items by Lindsjorn et al. [16] on topics of work satisfaction and learning (adapted and filtered), as well as the opinion of developers gathered in this work, the following questionnaire items were compiled:

1. I feel adequately compensated for my work.
2. I gain something from the collaborative teamwork.
3. I am able to acquire important know-how through collaboration.
4. The collaboration within my team promotes me personally.
5. The collaboration within my team promotes me professionally.

7.16 Shared Ideals

Shared Ideals is the result of feedback based on the survey study, rather than being based on literature. Therefore, the questionnaire items were constructed based on developer feedback without any literature reference:

1. I agree with the mission of the team and project.
2. I am happy with the direction of the project.
3. I am satisfied with the way the team is being led and organized.
4. The frequency and degree of collaboration within the team is adequate.

7.17 Engagement

An established measurement scale for general work engagement is the *"Work Engagement Scale"* by Utrecht et al. [25]. The authors categorize engagement in three categories: Vigor, dedication and absorption. However, many of the questionnaire items are not well suited for the application within this quality model. A clear example of this are the first three measures for vigor: *"1. At my work, I feel bursting with energy 2. At my job, I feel strong and vigorous 3. When I get up in the morning, I feel like going to work"*. These items merely serve to assess the level of engagement of an individual and are not designed around improvable measures. Therefore, instead of adapting the whole scale, select items describing general work engagement were adapted and supplemented with more specific items inspired by what software developers noted as important for their engagement:

1. I find the work that I do full of meaning and purpose.
2. I enjoy my work.
3. I am proud of the work that I do.
4. To me, my work is challenging.
5. I am confident in the success of the project.
6. I am satisfied with the amount of freedom I have in my work.
7. I experience frequent successes, such as meeting milestones or satisfying iterations of the software, with my team.
8. I feel like my personal qualities and achievements are acknowledged and appreciated by the organization.

Chapter 8

Review and future work

8.1 Conclusion

Within the scope of this work, a quality model for collaboration in software team was constructed based on literature, expert interviews and an online survey. It underwent multiple validation processes and revisions. The model was well received by practicing software developers, with the included quality aspects all being considered important for collaboration. Thus the model is deemed appropriate for the application in software teams. The quality model is an answer to both research questions formulated in the introduction of the thesis:

- **RQ1:** What constitutes high quality collaboration in software teams?
- **RQ2:** How can the quality of collaboration in software teams be measured and improved?

The constituents of high quality collaboration in software teams are represented through the model's quality aspects (cf. figure 6.1), answering the first research question. Improvable metrics for all of the constituents are presented in chapter 7, answering the second research question.

To the best of knowledge, this is the only quality model (as defined in chapter 2.1) for collaboration in software teams and furthermore is the only general model for collaboration which is both highly specific to software teams (including unique needs and constituents), while also not being over-fitted to any specific mode of software development.

The data gathered within this work further justified the existence of a quality model for collaboration, with many statements exemplifying the need for clearer distinctions of socio-technical aspects through the acknowledgment of heavy interdependencies between the individual factors.

8.2 Future Work

The clear next step for future work is the controlled and monitored application of the model in one or multiple software projects under the close guidance of a researcher. This was not possible in the scope of this work due to limited time and resources. While the model was validated by multiple software developers, only the real application in software teams can yield rich data about the effectiveness of the model. Analyzing collaboration in a software project with the help of the quality model would lead to definitive insights about the measurability and improvability of the constituents, as well as answering the question if the improvement of the constituents actually leads to a noticeable increase in quality of collaboration. Perhaps, this process could also lead to further inclusions or exclusions of quality aspects in the model.

Secondly, analyzing the exact interdependencies between the quality aspects (or subdimensions) could lead to new insights about influences and event chains in regards to collaboration. This could make problems with collaboration more analyzable and identifiable. While some sporadic statements were gathered about which aspects influence one another within this work, a more systematic approach (for example through paired comparison and *Pathfinder Algorithms*) is necessary to make definitive statements about the interdependencies.

Bibliography

[1] L. Augustin, D. Bressler, and G. Smith. Accelerating software development through collaboration. In *Proceedings of the 24th International Conference on Software Engineering*, pages 559–563, 2002.

[2] D. Batra, W. Xia, and M. Zhang. Collaboration in agile software development: Concept and dimensions. *Communications of the Association for Information Systems*, 41(1):20, 2017.

[3] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al. The agile manifesto, 2001.

[4] B. Boyd, A. Townsley, C. Walter, C. Johnson, and R. Gamble. Examining collaboration among student teams relying on web applications to coordinate software development. 2017.

[5] B. Braunschweig and C. Seaman. Measuring shared understanding in software project teams using pathfinder networks. In *Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement*, pages 1–10, 2014.

[6] B. Çaglayan and A. B. Bener. Effect of developer collaboration activity on software quality in two large scale projects. *Journal of Systems and Software*, 118:288–296, 2016.

[7] V. R. B. G. Caldiera and H. D. Rombach. The goal question metric approach. *Encyclopedia of software engineering*, pages 528–532, 1994.

[8] S. Cheah, S. Li, and Y.-P. Ho. Mutual support, role breadth self-efficacy, and sustainable job performance of workers in young firms. *Sustainability*, 11(12):3333, 2019.

[9] A. C. Costa and N. Anderson. Measuring trust in teams: Development and validation of a multifaceted measure of formative and reflective indicators of team trust. *European Journal of Work and Organizational Psychology*, 20(1):119–154, 2011.

[10] D. Ford, M.-A. Storey, T. Zimmermann, C. Bird, S. Jaffe, C. Maddila, J. L. Butler, B. Houck, and N. Nagappan. A tale of two cities: Software developers working from home during the covid-19 pandemic. *ACM Trans. Softw. Eng. Methodol.*, 31(2), dec 2021.

[11] S. L. Grover. Unraveling respect in organization studies. *Human relations*, 67(1):27–51, 2014.

[12] M. Hoegl and H. G. Gemuenden. Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence. *Organization science*, 12(4):435–449, 2001.

[13] International Organization for Standardization. ISO 25010, software and data quality. `https://iso25000.com/index.php/en/iso-25000-standards/iso-25010`. Accessed: 24.06.2024.

[14] J. Klünder, O. Karras, N. Prenner, and K. Schneider. Modeling and analyzing information flow in development teams as a pipe system. In *MoDELS (Workshops)*, pages 730–737, 2018.

[15] F. Lanubile. Collaboration in distributed software development. In *International Summer School on Software Engineering*, pages 174–193. Springer, 2006.

[16] Y. Lindsjørn, D. I. Sjøberg, T. Dingsøyr, G. R. Bergersen, and T. Dybå. Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software*, 122:274–286, 2016.

[17] R. C. Mayer, J. H. Davis, and F. D. Schoorman. An integrative model of organizational trust. *Academy of management review*, 20(3):709–734, 1995.

[18] B. McEvily and M. Tortoriello. Measuring trust in organisational research: Review and recommendations. *Journal of Trust research*, 1(1):23–63, 2011.

[19] A. Meneely and L. Williams. Socio-technical developer networks: Should we trust our measurements? In *Proceedings of the 33rd international conference on software engineering*, pages 281–290, 2011.

[20] B. Mullen and C. Copper. The relation between group cohesiveness and performance: An integration. *Psychological bulletin*, 115(2):210, 1994.

[21] M. Paasivaara and C. Lassenius. Collaboration practices in global inter-organizational software development projects. *Software Process: Improvement and Practice*, 8(4):183–199, 2003.

[22] E. Salas, D. E. Sims, and C. S. Burke. Is there a "big five" in teamwork? *Small group research*, 36(5):555–599, 2005.

[23] D. Samadhiya, S.-H. Wang, and D. Chen. Quality models: Role and value in software engineering. In *2010 2nd International Conference on Software Technology and Engineering*, volume 1, pages V1–320–V1–324, 2010.

[24] D. Sasse. Master's thesis: Interviewstudie zu sozialen aspekten in modernen softwareprojekten, 2023.

[25] W. B. Schaufeli, A. B. Bakker, and M. Salanova. Utrecht work engagement scale-9. *Educational and Psychological Measurement*, 2003.

[26] B. Singh and S. P. Kannojia. A review on software quality models. In *2013 International Conference on Communication Systems and Network Technologies*, pages 801–806. IEEE, 2013.

[27] D. Strode, T. Dingsøyr, and Y. Lindsjorn. A teamwork effectiveness model for agile software development. *Empirical Software Engineering*, 27(2):56, 2022.

[28] E. Weimar, A. Nugroho, J. Visser, A. Plaat, M. Goudbeek, and A. P. Schouten. The influence of teamwork quality on software team performance. *arXiv preprint arXiv:1701.06146*, 2017.

[29] J. Whitehead. Collaboration in software engineering: A roadmap. In *Future of Software Engineering (FOSE'07)*, pages 214–225. IEEE, 2007.