

Non-Technical Individual Skills are Weakly Connected to the Maturity of Agile Practices

Lucas Gren^{a,*}, Alessia Knauss^a, Christoph Johann Stettina^{b,c}

^aChalmers University of Technology and the University of Gothenburg, SE-412 96 Gothenburg, Sweden

^bCentre for Innovation, Leiden University, Schouwburgstraat 2, 2511 VA, The Hague, The Netherlands

^cLIACS, Leiden University, Niels Bohrweg 1, 2333 CA, Leiden, The Netherlands

Abstract

Context Existing knowledge in agile software development suggests that individual competency (e.g. skills) is a critical success factor for agile projects. While assuming that technical skills are important for every kind of software development project, many researchers suggest that non-technical individual skills are especially important in agile software development.

Objective In this paper, we investigate whether non-technical individual skills can predict the use of agile practices.

Method Through creating a set of multiple linear regression models using a total of 113 participants from agile teams in six software development organizations from The Netherlands and Brazil, we analyzed the predictive power of non-technical individual skills in relation to agile practices.

Results The results show that there is surprisingly low power in using non-technical individual skills to predict (i.e. explain variance in) the mature use of agile practices in software development.

Conclusions Therefore, we conclude that looking at non-technical individual skills is not the optimal level of analysis when trying to understand, and explain, the mature use of agile practices in the software development context. We argue that it is more important to focus on the non-technical skills as a team-level capacity instead of assuring that all individuals possess such skills when understanding the use of the agile practices.

Keywords: skills, agile practices, code quality, empirical study

1. Introduction

Agile methods are increasingly used in industry as they are established to support projects in their success [55]. Cockburn & Highsmith [10] argue that individual competency is an important success factor in agile projects. In agile methods “the emphasis on people and their talent, skill, and knowledge becomes evident.” Even on team-level, they argue that the emphasis is again “on competency rather than process.” Literature suggests that we progress through two major stages during the development of a cognitive skill, a declarative knowledge stage and a procedural knowledge stage [3]. While the former can be acquired by reading text books (e.g. learn how to lead a team), the latter, the procedural knowledge, can only be acquired in process (e.g. by actually leading a team and learning from mistakes). Hence, it seems that success in agile projects depends on individual skills, that are developed in individuals over time. Many studies in software engineering have focused on explaining the individual skills (see e.g. Turley & Bieman [66]), which implies that the individual non-technical skills are believed to predict team-level performance in relation to collaborative aspects. However, no studies have looked at explicit connections between agile

*Corresponding author. Tel.: +46 739 882 010

Email addresses: lucas.gren@cse.gu.se (Lucas Gren), alessia.knauss@chalmers.se (Alessia Knauss), c.j.stettina@cdh.leidenuniv.nl (Christoph Johann Stettina)

practices and individual non-technical skills. As we assume technical skills to always be a precondition for a successful software development endeavour, non-technical skills seem to be important in such endeavours as well [5]. Such non-technical individual skills have been stated as especially important in agile practices, since they focus more on individuals than processes [10].

Therefore, this paper investigates the connections between (i.e. the predictive power of) thirteen self-assessed non-technical individual skills and the intended mature use of a set of eight common agile practices. In this paper, we define *practice maturity* by assuming it to be possible to describe by an evolutionist model comprising of a progressive and directional set of changes. As the practices mature, they increase in perfection or complexity over time [33]. We assume the end goal of the development of the agile practices to be to what degree a practice is implemented in its intended way, as measured by the degree of agreement to items in the *perceptive agile measurement* [63]. We did not investigate the use of agile practices by having subjects tick the practices they use from a list, but instead used the perceptive agile measurement [63] to assess the degree of agile behavior prescribed by an agile practice. We assume different degrees of such agile behavior to be equivalent to different levels of agile practice maturity. The perceptive agile measurement was created to assess the social-psychological effect of agile practices, which provides a higher resolution of the actual behaviour shown in these agile teams.

The sample consisted of agile team members from seven organizations in Brazil and The Netherlands. We set up eight hypotheses regarding the associations between all individual skills and each agile practice. In order to assess the hypotheses, we conducted a diversity of analyses: First, we checked our survey data for normality. Second, we ran eight independent ANOVAs. Third, we built three new regression models for the significant models and analyzed their effect sizes.

The results show non-significant or negligible effect sizes for all our analyses. We therefore conclude that looking at non-technical individual skills is not the optimal level of analysis when wanting to understand the use of agile practices, and argue for using the “agile team” as the level of analysis instead.

To clarify the outline of this paper, the next section (Section 2) presents related work with regards to individual and team level skills in agile software development. Section 3 gives an overview of the measurements and constructs (i.e., non-technical individual skills and agile practice) used in this paper. Section 4 depicts the method used to measure agility and non-technical individual skills and correlate these measurements in a set of multiple linear regression models, Section 5 presents the results, which are discussed in Section 6, followed by conclusions and future work in Section 7.

2. Related Work

In this section, we provide a definition of what skills are, provide an overview of research on individual skills in the software engineering context, and present the few studies looking at a higher level of analysis in relation to skills in the software engineering domain.

2.1. What Is a Skill?

We define a skill as “an ability to do something well; expertise” [60] in this study in its broader sense. However, in order to understand what this means, we need to look at related research in what characterizes a skill and how they are acquired. As mentioned in the introduction, the acquisition of a cognitive skill can be described by two major stages: a declarative knowledge stage and a procedural knowledge stage [3]. These stages are intimately connected to what Argyris [4] calls single- and double-loop learning. We can learn to repeat new information without integrating it on a deeper level. With such shallow learning, we can not apply our acquired knowledge in new situations and we fail to translate the new knowledge to similar cases [4]. Therefore, we asked the participants of this study about their own satisfaction in relation to specific skills, which then are more in relation to the procedural knowledge (double-loop) rather than shallower declarative (single-loop) knowledge in their work context. In this study, we also define a team-level skill as the capability of the entire team in relation to non-technical skills, i.e. even if all members can not plan well for themselves, the team as an entity might have good planning skills anyways as a product of collaboration and reciprocal help.

Another scientific field that has gone through similar phases in research is the medical field, and Fletcher et al. [21] shows how surgical teams have gone from only focusing on technical skills to also realizing the importance of the non-technical skills needed for successful treatment of patients. Fletcher et al. [21] divides the non-technical skills into two categories; (1) cognitive and mental skills (e.g. decision-making, planning, etc.) and (2) social or interpersonal skills (e.g. team-working, communication, leadership, etc.). This research highlights the importance of individual non-technical skills, which has also been common in the software engineering research and is presented next.

2.2. Research on Individual Skills in Agile Software Development

Agile methods are described as having a strong focus on individuals and their skills [10]. Cockburn & Highsmith [10] conclude that individual skills seem more important than team characteristics – even on the team level they argue that the emphasis is on individual competency. Strengths as well as weaknesses of individual skills need to be taken into consideration as both can have an influence on the success of an organization. Conboy et al. [14] report an increased exposure of capabilities and reliance on social skills. They report that exposing weaknesses of team members can often be counter-productive and even highly respected and performing team members can be bullied, challenging existing organizations. Furthermore, an understanding of the skills necessary within a team can help team members in their development.

In agile software development, the individual skills of software engineers need to be considered, according to many studies. For example, Turley & Bieman [66] identified 38 essential competencies of software engineers. Among the top ten are: (1) Team Oriented, (2) Seeks Help, (3) Helps Others, (4) Use of Prototypes, (5) Writes/Automates Tests with Code, (6) Knowledge, (7) Obtains Necessary Training/Learning, (8) Leverages/Reuses Code, (9) Communication/Uses Structured Techniques for Communication, and (10) Methodical Problem Solving. In addition to the general skills that a software engineer should possess, skills of requirements engineers play an important role in agile software development. The focus in agile software development is on the customer who decides on what is of value and who is supposed to be on site to clarify requirements [22]. Hence, contrary to traditional RE, in agile development any member from the development team can directly interact with the customer and collect requirements [59]. Furthermore, RE represents an area in which it is especially important to consider social/non-technical skills and theories [67]. Hence, non-technical skills of requirements engineers are important and required for each individual team member in agile software development.

Negotiation is an inevitable element in RE [47, 24]. For example in requirements elicitation and analysis it plays a major role as it supports handling stakeholder conflicts concerning requirements [1]. Negotiation promotes a shared vision, shared knowledge, and cooperation among stakeholders [1]. Furthermore, communication is an inevitable skill in RE [15]. Team members need to communicate with each other for many reasons, i.e. bugs discussion, code issues, code reviews, code refactoring, code synchronization, coordination, management, support issues, sprint planning, quality, user story clarification and user story negotiation [30]. In agile requirements engineering most activities depend on communication between different parties, their input and judgment. Hence, it is highly dependent on the skills of the team members. Further skills for phased as well as agile RE are: 1) Dividing bigger tasks into small ones, 2) giving up control – as code is developed and changes by different people, 3) writing meaningful tests, 4) conversation, 5) object-oriented design, 6) fast cycle times [35].

2.3. Research on Team-Level Skills in Agile Software Development

Most existing research focuses on individual skills, as presented in Section 2.2. A few studies also report about the importance of non-technical skills on a team level (e.g., with regards to team work and setting up teams). Lalsing et al. [37] identified the underlying people factors for a team to be effective in agile software development, and several agile projects were in the study. Results showed that for projects exceeding the project budget, issues were e.g. related to team communication and collaboration (i.e. trust and interaction). They concluded that it is crucial to select the “right people for the right team,” and not only the “right people.” Tanner & von Wilming [65] study success and failure of agile projects in waterfall environments. From studying two cases of agile projects and literature on this topic, they concluded that the following

factors have an influence on the projects success or failure of agile projects: 1) Culture, 2) Customer Involvement and Mandate, 3) Stakeholder Involvement and Buy-In, 4) Team Structure and Team Logistics, 5) Project Type and Project Planning, and 6) Skill Level and Attitude of Team Members. Crowder & Friess [17] identified communication, coordination, trust and team orientation as the most important team factors for distributed agile teams, based on survey data. From a systematic literature review, they identified team orientation, shared leadership, mutual performance monitoring, backup behavior, feedback, team autonomy, team learning, coordination, communication, trust, collective culture, ease of use of technology, and team familiarity as teamwork factors. Gren et al. [23] also found that team maturity (from a group dynamics perspective) is a key factor in the success of building agile teams in large organizations.

A challenge when trying to understand productivity in general is to get data from the right level of analysis [25]. Software engineering research on human factors can learn a great deal from the journey social psychology has done from the individual to the group as a level of analysis (cf. Hogg & Williams [27]), a journey that has largely been repeated by organizational science [34]. If we want to understand agile practices, the individual skills level might be too much on a micro level based on research in other fields, and we could study the agile teams and how software developers and other team-members behave collectively instead, i.e. explained variance in data might come from the meso level of the team as an entity. Recent research in social psychology [72, 19] has shown that, in general, the performance of teams on a diversity of tasks is set on group-level independent of the intelligence of the individuals. The intelligence of groups have instead been shown to be more dependent of social sensitivity (i.e. a person’s ability to read emotions in facial expression), and conversational turn-taking (i.e. groups were less collectively intelligent if a few individuals dominated the conversations). Such findings contradict the conclusions drawn in a lot of software engineering research on the importance of non-technical individual skills (see for example Turley & Bieman [66]). To clarify such contradictions, we conduct a study in the software engineering domain aiming to shed light on this contradiction of individual vs. group level. As software engineering mainly looks at individual level, this is the level we chose to investigate in this current study. According to Hackman [25], one should preferably look at the macro (i.e. organizational) level in addition to the micro and meso levels. While we recognize that variance possibly could be explained on the organizational level, we focus on discussing the micro and meso levels in this study.

3. Measurements, Constructs, and Research Hypotheses

In this section we describe the measurements used in the current study and their operationalization based on previous studies. We first present theory on common agile practices and how they can be measured, and then we suggest a measurement of self-assessed non-technical individual skills based on such studies in the agile context.

3.1. Common Agile Practices

Agility as a concept can be difficult to delineate [36]. Agility emerged in practice and has been discussed across different scientific domains such as manufacturing and logistics [6], business management [48], information systems literature [13], and sports science [58], which makes it difficult to define. Fuelled by the difficulties in definition, an analysis by Laanti et al. [36] suggests to look at agility as a set of concrete practices to understand agility. Other studies have followed that track and investigated the usage and perceptions of practices perceived as agile within software development teams [70, 63]. In line with Salvato [53] we believe that such concrete, routinized activities have a huge impact on the effectiveness and sustainability of project management processes – and, as such, that there must be some kind of behaviour that can be considered “more” agile than other behaviour connected to more traditional project management groups. Early Scrum literature generally describes to the following practices [54]: (1) Collocated scrum teams, (2) Daily scrum stand-up meetings (3) Iteration planning in sprint planning meetings, (4) Iterative development in sprints, and (5) Sprint reviews.

Based on previous research and perceptions of practitioners, So & Scholl [63] constructed an instrument for quantitative analyses of social-psychological effects of the above mentioned agile practices, however

categorized a bit differently than those Scrum practices above. They created scales for the eight of the core agile practices, namely (1) Iteration planning, (2) Iterative development, (3) Continuous integration and testing, (4) Stand-up meetings, (5) Customer access, (6) Customer acceptance tests, (7) Retrospectives, and (8) Collocation. The framework of So & Scholl [63] is particularly useful as they provide for the first time a scientifically validated psychometric instrument covering these eight core agile practices, i.e. their tool tries to capture agile behaviour in connection to the practices and therefore claims to measure the actual practices. The measured practices Customer access, and Customer acceptance tests, also extend their survey to include aspects of agile requirements engineering, which adds a focus on the entire development chain from planning to delivering in the agile context.

We use the questionnaire suggested by So & Scholl [63] to understand the connection of agile practices and individual skills with the reason that we need social-psychological measurements in our study. In the following, we will elaborate on the eight practices included in the instrument by So & Scholl [63]. Furthermore, in the text boxes below each of the depicted practice we reproduce the exact items (i.e., the agile practice and sub-questions used to cover this agile practice) used in our questionnaire.

Iteration Planning. In a Collaborative planning workshop the deliverables and scope of an iteration is defined with all team members being present. It is executed at the beginning of each iteration, sometimes also referred to as *Planning Game*. The practice generally consists of two stages: In the first stage, requirements are gathered in the form of user stories to serve as a medium for discussions between the customer and the developers. In the second stage, the stories are revised, estimated and prioritized into an iteration backlog [41]. The active participation of technical team members in definition as well as estimation of user stories is considered as an indication for a mature application of the practice [69, 56].

Iteration Planning: (1) All members of the technical team actively participated during iteration planning meetings. (2) All technical team members took part in defining the effort estimates for requirements of the current iteration. (3) When effort estimates differed, the technical team members discussed their underlying assumption. (4) All concerns from team members about reaching the iteration goals were considered. (5) The effort estimates for the iteration scope items were modified only by the technical team members. (6) Each developer signed up for tasks on a completely voluntary basis. (7) The customer picked the priority of the requirements in the iteration plan.

Iterative Development. Routinized delivery of sub-results (working software) in short and iterations of fixed length [50]. Although the practice has been popularized with the dawn of agile methods, the application of iterative software development dates as far back as the mid-1950s [38]. Short iterations of 30 days or less together with continuous integration, have been found as the two practices considered most essential for a team to be considered agile [70]. Iterative development is a shared practice in agile methods as well as in user-centered design [7].

Iterative Development: (1) We implemented our code in short iterations. (2) The team rather reduced the scope than delayed the deadline. (3) When the scope could not be implemented due to constraints, the team held active discussions on re-prioritization with the customer on what to finish within the iteration. (4) We kept the iteration deadlines. (5) At the end of an iteration, we delivered a potentially shippable product. (6) The software delivered at iteration end always met quality requirements of production code. (7) Working software was the primary measure for project progress.

Continuous Integration and Testing. Holck & Jørgensen [28] define continuous integration as follows: (1) access of development team members to add contributions to the development version at any time, and (2) obligation of team members to integrate their own contributions properly. In order to enable such a continuous integration of ongoing development into a software system, the practices are often linked to (automated) testing methods to enable a timely verification of the system [26].

Continuous Integration and Testing: (1) The team integrated continuously. (2) Developers had the most recent version of code available. (3) Code was checked in quickly to avoid code synchronization/integration hassles... (4) The implemented code was written to pass the test case. (5) New code was written with unit tests covering its main functionality. (6) Automated unit

tests sufficiently covered all critical parts of the production code. (7) For detecting bugs, test reports from automated unit tests were systematically used to capture the bugs. (8) All unit tests were run and passed when a task was finished and before checking in and integrating. (9) There were enough unit tests and automated system tests to allow developers to safely change any code.

Stand-up meetings. Frequent team coordination meetings in which team members provide a status update to their colleagues. The meetings are generally hold standing up and are time boxed to 5-15 minutes to frame its short and focused nature. Each coaching session starts with a team stand-up where each group was asked the three common questions: “What have you done since the last meeting?” “What are you planning on doing until the next meeting?” and “What issues and impediments are you facing that prevent you from accomplishing these things?” An ethnographic account of the practice is provided by Sharp & Robinson [56]. Stray et al. [64] investigated the application of daily stand-up team coordination meetings. They found that only 24% of each of the meetings they studied focused on coordination. Rather, 35% of the meeting time was spent on content-discussions elaborating problem issues and discussing possible solutions.

Stand-Up Meetings: (1) Stand up meetings were extremely short (max. 15 minutes). (2) Stand up meetings were to the point, focusing only on what had been done and needed to be done on that day. (3) All relevant technical issues or organizational impediments came up in the stand up meetings. (4) Stand up meetings provided the quickest way to notify other team members about problems. (5) When people reported problems in the stand up meetings, team members offered to help instantly.

Customer Access. Availability of customers for product feedback and clarification of requirements is integral to effective agile teams and has been found as one of the critical success factors when implementing agile methods [45, 9]. Especially when moving away from traditional software development customer access can be a challenge as customers might not be used to close interaction [45].

Customer Access: (1) The customer was reachable. (2) The developers could contact the customer directly or through a customer contact person without any bureaucratic hurdles. (3) The developers had responses from the customer in a timely manner. (4) The feedback from the customer was clear and clarified his requirements or open issues to the developers.

Customer Acceptance Tests. Acceptance tests defined by the customer present a means to the developers to determine which iteration goals have been achieved at the end of each iteration [62].

Customer Acceptance Tests: (1) How often did you apply customer acceptance tests? (2) A requirement was not regarded as finished until its acceptance tests (with the customer) had passed. (3) Customer acceptance tests were used as the ultimate way to verify system functionality and customer requirements. (4) The customer provided a comprehensive set of test criteria for customer acceptance. (5) The customer focused primarily on customer acceptance tests to determine what had been accomplished at the end of an iteration.

Retrospectives. Workshop at the end of each iteration to improve the process and incorporate successful practices for the next iteration. In order to enable continuous improvement of the practices applied each team member lists “What went well” and “What could be improved” [42]. The impact, however, dependent largely on their implementation [44]. For example, retrospectives should take place at the end of each iteration and systematically assign all improvement points to responsible individuals [62].

Retrospectives: (1) How often did you apply retrospectives? (2) All team members actively participated in gathering lessons learned in the retrospectives. (3) The retrospectives helped us become aware of what we did well in the past iteration(s). (4) The retrospectives helped us become aware of what we should improve in the upcoming iteration(s). (5) In the retrospectives (or shortly afterward), we systematically assigned all important points for improvement to responsible individuals. (6) Our team followed up intensively on the progress of each improvement point elaborated in a retrospective.

Table 1: Non-technical individual skills used in the present study

Non-technical skills	Studies including skill
(1) Communication Skills	[39, 66, 15, 37, 32, 46, 17]
(2) Teamwork Skills	[39, 66, 10, 65, 46, 17]
(3) Collaboration Skills	[66, 37, 17]
(4) Ability to Meet Project Goals	[66, 8, 17]
(5) Customer Orientation	[66, 22, 32, 46, 65]
(6) Requirements Management Skills	[59, 1]
(7) Planning Skills	[39, 66, 46, 65]
(8) Leadership Skills	[39, 32, 46, 17, 12]
(9) Decision-Making Skills	[46, 31]
(10) Business-Minded Skills	[66, 17]
(11) Problem-Solving Skills	[66, 46]
(12) Organizing Skills	[39, 66]
(13) Negotiation Skills	[66, 24, 1, 57, 17]

Collocation. Close proximity of development team members is reported as one of the critical success factors when implementing agile methods [40]. It is recognized as one of the important vehicles for successful communication and knowledge creation [45].

Collocation: (1) Developers were located majorly in... (2) All members of the technical team (including QA engineers, db admins) were located in... (3) Requirements engineers were located with developers in... (4) The project/release manager worked with the developers in... (5) The customer was located with the developers in...

3.2. Common Non-Technical Skills

We use common denominators on skills derived from previous studies as presented in Section 2.2. In that analysis, we found 13 non-technical individual skills that have been identified in at least two papers as important for software developers to be successful. Table 1 presents the non-technical skills used in this study and supporting literature depicting the importance of this skill.

3.3. Research Hypotheses

Based on the literature, in this current paper, we investigate the assumption that the use of agile practices is positively connected to non-technical individual skills. All the items were self-assessed by team members in agile software development projects. The hypotheses were therefore the following:

- **H₁:** The mature use of the agile practice Iteration Planning is positively associated with agile team members' self-assessed non-technical skills.
- **H₂:** The mature use of the agile practice Iterative Development is positively associated with agile team members' self-assessed non-technical skills.
- **H₃:** The mature use of the agile practice Continuous Integration and Testing is positively associated with agile team members' self-assessed non-technical skills.

- **H₄:** The mature use of the agile practice Stand-Up Meetings is positively associated with agile team members' self-assessed non-technical skills.
- **H₅:** The mature use of the agile practice Customer Access is positively associated with agile team members' self-assessed non-technical skills.
- **H₆:** The mature use of the agile practice Customer Acceptance Tests is positively associated with agile team members' self-assessed non-technical skills.
- **H₇:** The mature use of the agile practice Retrospectives is positively associated with agile team members' self-assessed non-technical skills.
- **H₈:** The mature use of the agile practice Collocation is positively associated with agile team members' self-assessed non-technical skills.

4. Method

In order to investigate the connections between the individual skills and agile practices, we created a survey using the agile practices suggested by So & Scholl [63] and our aggregation of non-technical individual skills (both derived in the previous section).

4.1. Participants

All the participants were members of agile teams in the participating companies (as stated by our company contacts). We explicitly asked all team members to answer the survey, but to skip questions they were not able to assess. As a consequence, most respondents were intimately connected to development code in one way or the other. The reason why we did not ask specifically for only software developers, were that some employees that conduct software development might have another title, like for example “system engineer.”

This study was carried out at six organizations in total, three companies in Brazil and three in The Netherlands (two companies and one public sector IT department). These companies were selected because the first and third authors had direct or indirect research connections to people within the organizations. We wanted the participating organizations to be as diverse as possible in order to being able to generalize our study to the broader population of agile team members in the software development context, i.e. we intended to survey agile team members from different continents, different sizes of organization, as well as the public and private sectors (see Table 2).

Table 2: The Participating Organizations

Organization	Country	# of employees	# of responses
Company 1	Brazil	100	22
Company 2	Brazil	5,000	57
Company 3	Brazil	35	11
Company 4	The Netherlands	3,500	4
Company 5	The Netherlands	50,000	7
Public Sector Organization	The Netherlands	5,000	12
			Total: 113*

*After list-wise deletion due to missing values, we had at least 99 valid responses to use in our regression analyses.

The Brazilian sub-sample contained data points from IT departments at a large on-line media and social networking enterprise with around 5,000 employees, a smaller software consultancy company with around 35 employees, and a company that provides programming courses to individuals and companies, with around 100 employees. All the agile team members received the surveys via their managers but their replies were anonymous. The response rate was 92% for the Brazilian sub-sample due to the fact that they were collected on-site in paper form.

The Dutch sub-sample consisted of four groups across three organizations: an IT service provider (around 3,500 employees), banking and financial services (around 50,000 employees), and an IT service department in the public sector (around 5,000 employees). The response rate for the Dutch sub-sample was 81%.

All the participating companies said they use an agile development approach in the software development conducted, but with teams of different maturity levels in that process. The total number of respondents in the first measurement was 158 agile team members.

4.2. Survey items

In the survey these skills were separately put as questions in the following form: “How satisfied are you with your [skill]?” The reason why we used personal satisfaction of a certain non-technical skill was that, in agile teams, such skills are perceived as utterly important. Therefore, a satisfaction of a skill should be more related to a perceived peer-evaluation by a team-member than if asked to only rate their own individual skills. In addition, as mentioned previously, we aimed at investigating procedural rather than declarative skills, and in order for a participant to rate their individual skill high in context, we believe such rating will be more in relation to their declarative knowledge.

We used the survey suggested by So & Scholl [63] to measure the mature use of the eight agile practices included in their study, i.e. higher scores on their survey imply higher maturity of a practice since the questions are in relation to the intended, and therefore mature, use of a practice. The entire construct used is presented in Section 3. Due to all the different definitions and ambiguity of “agility” [36], we chose this survey since it captures the social-psychological behavior in connection to what the different practices try to achieve. It is also the only tool we have found that is validated through a factor analysis [20] and a reliability analysis (using the Cronbach’s α [16]) with a sample of $N = 227$ [63].

The agile items in the questionnaire were assessed on a 7-point Likert scale (1 = never and 7 = always), with one exception being the Collocation items that were rated from 1 = the same room to 5 = different timezones. These scales were used for the simple reason that these measurements were developed and validated using those exact scales. The skills were assessed from 1 = completely dissatisfied and 9 = completely satisfied.

In order to validate our results further, we also built a regression model using perceived code quality as a response variable. The perceived code quality aspect was measured using the single question: “How would you rate the code quality in your product(s)?” rated from 1 (very poor) to 4 (excellent). The reason for this validation was to investigate if non-technical individual skills were connected to a completely different aspect of the software developed (i.e. other than the agile practices). If that were the case, we would have support for the usefulness of measuring non-technical individual skills outside the scope of agile practices.

4.3. Data collection and analysis

The questionnaires were distributed in paper form by one of the managers and collected on site by the first and third authors (depending on the country).

To evaluate if the data was normally distributed, we plotted frequency histograms for all the multiple linear regression models (for one example, see Figure 1). Figure 1 shows that the residuals are randomly scattered around the regression line. However, we saw some issues with the dependent variable Retrospectives, which turned out to have a set of outliers. When these outliers were removed and the data looked normally distributed, the ANOVA for that category was still not significant, meaning that the outliers did not affect the result. We also checked the variance inflation factor (VIF) for each regression model and all values were below 3, which is acceptable for these kinds of analysis since a common rule of thumb is below 10 [43].

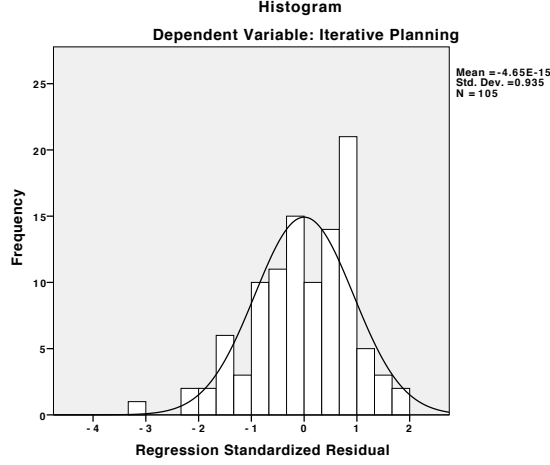


Figure 1: Frequency histogram with Iteration Planning as dependent variable and all the individual skills as factors.

In order to investigate the connections between the two concepts we first ran eight ANOVAs with all the skills as factors and the agile practices as response variables one by one. The purpose was to investigate the predictive power of knowing the agile team members perceived individual skills on all the agile practices separately (plus the quality question in its own analysis as a validation). This means that we investigated how much of the variance in a measured agile practice that was explained by the non-technical individual skills together, and therefore, opted to use linear regression analysis with all the skills as factors. It is important to note the differences between predictive and causal models and in this study we only claim the former. However, since we have the theoretical assumption that skills predict agile practices maturity and not the other way around, we have the skills as independent variables and the agile practices measurements as dependent variables. If the ANOVA was significant at an alpha level of 5%, we proceeded and built a multiple linear regression model to see which skills (factors) were significant. As a measurement of effect size we used η^2 (often called R^2 in regression analysis) for each omnibus test (i.e. ANOVA) [11].

5. Results

To assess the predictive power of non-technical individual skills on the agile practices, we ran eight independent ANOVA omnibus tests of which only the three response variables “Iteration Planning” ($F = 2.166, p = 0.017, N = 105$), “Customer Access” ($F = 2.415, p = 0.008, N = 102$), and “Customer Acceptance Tests” ($F = 2.940, p = 0.001, N = 99$) were significant. All the other ANOVAs using “Iterative Development” ($F = 1.307, p = 0.224, N = 103$), “Continuous Integration and Testing” ($F = 0.664, p = 0.792, N = 100$), “Stand-Up Meetings” ($F = 0.862, p = 0.595, N = 99$), “Retrospectives” ($F = 0.946, p = 0.510, N = 101$), and “Collocation” ($F = 0.968, p = 0.488, N = 103$) as dependent variables were not significant at an alpha level of 5%. We therefore failed to reject the null-hypotheses in favour of H_2, H_3, H_4, H_7, H_8 using the agile practices as response variables, i.e. they were not significant at an alpha level of 5%. Therefore, we also conclude that we have weak support, but still reject the null hypotheses in favour of H_1, H_5 , and H_6 . Next, we explain why the support was weak.

The agile practices Iteration Planning, Customer Access, and Customer Acceptance Tests had significant ANOVAs and we therefore ran further analyses using the significant factors (i.e. non-technical individual skills) in order to evaluate the size of the effects found. For these three significant omnibus tests, we built new models based on the significant factors and calculated effect sizes, which were found to be low or very low. The results were: Iteration Planning (adjusted $R^2 = 11.7\%$), using planning and teamwork skills as factors (see Table 3), Customer Access (adjusted $R^2 = 6.0\%$), using business-minded skills and organizing skills as factors (see Table 4), and finally Customer Acceptance Tests (adjusted $R^2 = 4.6\%$), using organizing skills

Table 3: Linear Regression Coefficients (Dependent Variable: Iteration Planning with 112 valid cases). Adjusted $R^2 = 11.7\%$

Model	Unstandardized B	Std. Error	Standardized B	t	p-value
(Constant)	3.373	0.528		6.389	0.000*
Teamwork Skills	0.175	0.074	0.778	2.354	0.020*
Planning Skills	0.133	0.062	0.208	2.148	0.034*

*p<.05

Table 4: Linear Regression Coefficients (Dependent Variable: Customer Access with 108 valid cases). Adjusted $R^2 = 6.0\%$

Model	Unstandardized B	Std. Error	Standardized B	t	p-value
(Constant)	5.251	0.692		7.590	0.000*
Business-Minded Skills	0.173	0.080	0.204	2.150	0.034*
Organizing Skills	-0.181	0.077	-0.222	-2.342	0.021*

*p<.05

as a factor (see Table 5). The only model with explained variance over ten percent was predicting “Iteration Planning” by using planning and teamwork skills. The regression models built using Customer Access and Customer Acceptance Tests had effect sizes under 10%, showing low predictive power of non-technical individual skills, even though we rejected the null hypotheses.

Higher teamwork and planning skills were connected to better iteration planning, which is the only result that makes sense. Actually, organizing skills were negatively correlated to Customer Access and Customer Acceptance Tests, which questions the relevance of the results in general. This would then mean that good organizing skills would be bad for mature Customer Access and Customer Acceptance Tests practice, which seems odd. However, we believe these results also point out that individual satisfaction of e.g. organizing skills is a poor predictor of agile maturity of a practice. Additionally, these effect sizes should be seen as irrelevant according to Cohen [11], while above ten percent is considered only a small effect. Therefore, the significant connections between teamwork and planning skills and iteration planning are the only ones relevant to analyze further.

The variance in the measurement of Iteration Planning, could be explained by 11.7%, which is considered a small effect in these types of studies [11]. As mentioned before, it makes sense that the individual skills of teamwork and planning would be connected to how well a team plans for an iteration, however, it makes equally much sense that individual communication, collaboration, decision-making, problem-solving, organizing, and negotiation skills would be connected to e.g. the team ability to develop iteratively. In addition, the effect was barely over ten per cent, which we still consider much lower than would be the case if all these agile team practices depended on individual non-technical skills.

Table 5: Linear Regression Coefficients (Dependent Variable: Customer Acceptance Tests with 104 valid cases). Adjusted $R^2 = 4.6\%$

Model	Unstandardized B	Std. Error	Standardized B	t	p-value
(Constant)	5.811	0.757		7.679	0.000*
Organizing Skills	-0.269	0.111	-0.234	-2.431	0.017*

*p<.05

Validating individual skills against perceived code quality. As a validation of the non-technical individual skills measurements we also built a model using the perceived code quality in products as a response variable. The non-technical individual skills showed no connection to the agile team members’ perceived code quality ($F = 1.172, p = 0.314, N = 99$), which means that non-technical individual skills also failed to predict the agile team members’ perceived quality of the code.

Summary of statistical results. To summarize the results above, we first analyzed our survey data for normality and plotted frequency histograms for multiple regression models using non-technical individual skills as factors and the agile practices (one-by-one), and perceived code quality, as response variables. To assess the predictive power of non-technical individual skills on the agile practice and perceived quality, we ran nine independent ANOVA omnibus tests of which only the three response variables “Iteration Planning,” “Customer Access,” and “Customer Acceptance Tests” were significant. For these three significant omnibus tests, we build new models based on the significant factors and calculate effect sizes, which were found to be low or very low. The only model with explained variance over ten percent was predicting “Iteration Planning” by using teamwork and planning skills. This result makes sense, however, so would many other predictions which were not significant, and in addition, higher organizing skills were connected to lower Customer Access and Customer Acceptance Tests, which questions the relevance of such a measurement. The data analyses therefore have shown that looking at non-technical individual skills is not the optimal level of analysis when wanting to predict agile maturity. What this means and the implications of the low predictive power of non-technical individual skills in connection agile practices will be discussed next.

It is important to, again, highlight the differences between correlation/predictive and causal models and in this study we only claim the former.

6. Discussion

The results show that there is very little predictive power when using self-assessed non-technical individual skills to understand the perceived maturity of the agile practices. Based on previous work in software engineering we would *not* expect such a result. This means that we can not look at self-assessed non-technical individual skills when trying to predict the intended and mature use of agile practices.

If there is little value in looking at individual non-technical skills when understanding or improving agile practices, what is then the option? As mentioned in the introduction, Turley & Bieman [66] identified 38 essential competencies of software engineers on different abstraction levels, which we interpret as an indication of the issues shown in our present study with using non-technical individual skills in order to increase the use of agile practices in software development teams, i.e. they are simply too many. In accordance with Tanner & von Wilingh [65] and Crowder & Friess [17] who investigated agile project success in relation to team orientation, shared leadership, backup behavior, feedback, team autonomy, team learning, coordination, communication, trust, collective culture, team familiarity, customer involvement and mandate, stakeholder involvement and buy-in, and team structure and team logistics, we have also found empirical support for looking at other levels of analysis than the individual, when wanting to optimize the benefits from an agile approach. Since project success is connected to agility [55] and So & Scholl [63] suggest a measurement for the intended use of the agile practices, we assume that higher scores on the agile practices measurement do imply a higher probability of project success. Lalsing et al. [37] also state that it is of utter importance to find the “right people for the right team” and not only the “right people,” which seems to be a key when building teams that can leverage agile practices in the way that they are intended.

Hackman [25] underlines the importance of crossing levels in organizational research and in this present study, we have shown that the individual level does not explain much variance and we instead need to investigate the team as the level of analysis. The cross-section between what is team and organization could be hard to define and e.g. organizational routines could be defined as both [49]. However, the distinctiveness between teams in organizations are often possible to find, and we argue the team-level needs to be in focus instead of the individual level, but preferably also in relation to the organizational level.

As a comparison, we looked at how Edum-Fotwe & McCaffer [18] present individual skills needed in the construction industries. They suggest skills not far from what is suggested in software development, which

supports our claim that these non-technical individual skills are too general to be useful in predictions of the dynamics of the specific organizational case. Edum-Fotwe & McCaffer [18] divide the needed abilities into primary and secondary knowledge and skill elements for developing project management competencies. Among the primary knowledge and skill elements, they report: (1) Planning and scheduling, (2) Construction management activities, (3) Basic technical knowledge in own field, (4) Productivity and cost control, (5) Leadership, (6) Delegation, (7) Negotiation, (8) Decision making, (9) Motivation and promotion, (10) Team working, (11) Time management, (12) Top management relations, (13) Establishing budgets, (14) Reporting systems, (15) Drafting contracts, (16) Communication skills Presentation, (17) General and business correspondence, (18) Report writing, (19) Chairing meetings, and (20) Understanding of organization.

We believe our findings imply that the teams need all the abilities that the non-technical individuals skills try to capture, but should be seen, and investigated, as a team capacity instead. This means that there is a difference between an individual having team working skills and the skills the team as a whole possesses. Our present study supports the findings presented in Section 2.3, that team skills are key to implementing and using agile practices. The collective intelligence is a property of the team itself [72] and, therefore, also the agile practices, i.e. just like the collective intelligence is unrelated to individual intelligence [72], individual non-technical skills seem to be unrelated to non-technical team skills. In addition, personalities can be consciously changed over time [29] and depend on our group membership [51]. From the software development context, Gren et al. [23] present an interesting quote in their study saying that the interviewees were surprised by how vocal some, previously very quiet, programmers get on some of their agile teams. Such a finding has extensive empirical support from social psychology and the studies of how context and its social expectations and interactions form reality [61].

6.1. Threats to Validity

We reflect on the threats to validity using internal, external, construct, and conclusion validity following the guidelines by Wohlin et al. [71].

Internal Validity. We only used the self-assessed (i.e. perceived) non-technical individual skills. It is difficult to determine the correlation of these agile team members’ self-assessed skills to their actual skills, or peer-assessed skills. However, it has been proven in psychology research that people overestimate their skills systematically (see e.g. Alicke et al. [2]). Hence, we assume our self-assessment to be a valid measurement when building associative models with relative associations between variables. A potential other threat is the operationalization of “skills” into asking about the agile team members’ satisfaction of their own skill. As mentioned in the method, the reason for using the personal satisfaction of a certain non-technical skills was the fact that, in agile teams, such skills are perceived as utterly important and a satisfaction of a skill should then be more related to a peer-evaluation than if asked to only rate their individual skills. Yet, we recognize that the reported individual skills could differ from “real” or the actual skills perceived by peers.

We did not include participants from companies not following agile practices for a comparison between other types of software development work practices. The rational behind this was our focus on companies using agile practices and the fact that we prioritized having a high number of participants than comparing the results to participants from organizations not following agile practices. We would also have had to specify and measure other work practices, which then became out of scope for this study. However, we would not be surprised if non-technical individual skills would turn out to be weakly connected to other types of high performance work practices, both in other software development methods, but also in other fields.

External validity. The sampling in this study represents a convenience sampling procedure, as the authors of this paper had direct or indirect research connections to people within the organizations that chose participants to the study. To mitigate this threat, we tried to diversify the sample of participants to include as a representative sampling of agile teams as possible. We involved participants from seven organizations from different continents, different sizes of organizations, as well as the public and private sectors. In addition, in order to correlate skills to levels of agile maturity, the organizations, and the participating agile teams, were all on different agile maturity levels, both within and between organizations. Four organizations

were based in Brazil and three organizations were from Europe, The Netherlands, including two companies and one public sector IT department. We therefore believe that our sample is representative for agile team members since we sampled from IT organizations of different types and sizes. Again, we aimed at looking for correlations, but allowing the skills measurements to co-vary, which makes such a sample appropriate. We also opted not to specify that we only wanted software developers to answer the survey, since we knew that many team members conducting software development work, might have other titles. We instead explicitly asked all members to respond, but told them to skip questions they could not assess. Therefore, our sample reflects all the agile team members that were involved in software development.

Conclusion validity. Due to our large sample size of 113 survey responses, we had a decent sample for building our linear regression models. We also made sure all the assumptions were fulfilled when building such statistical models.

Construct validity. To ensure that we do not just base our measurements on our assumptions, we conducted a literature review on individual skills in software development and agile projects, as well as on agile practices. From this review, we presented the most related work we found and step-by-step derived representative non-technical individual skills as well as agile practices. We only included a measurement if we found evidence for it in at least two publications. In addition, and more importantly, we failed to reject most of our hypothesis, which was not our initial intention, of course, but our data turned out to provide us with different results than we expected, i.e. we reported our results and discussed it without trying to fish for significant p values in data.

A common construct validity threat is hypothesis guessing. Since the participants filled out their surveys on paper and totally anonymous (i.e. not even stating their role or gender), we believe they were given a chance to answer honestly. We also did not inform participants about any of our hypotheses, but instead introduced our research topic of finding drivers behind agile practices in general over a large sample from more than one country.

7. Conclusions and Future Work

This paper set out to investigate the assumption that non-technical individual skills are positively connected to the mature use of agile practices. Through building a set of multiple linear regression models using a total of 113 survey responses, we analyzed the predictive power in measuring individual skills in relation to agile practices. We found that there is very low power in using non-technical individual skills to predict the maturity of agile practices in software development teams. We, therefore, conclude that looking at non-technical individual skills is not the optimal level of analysis when trying to understand and predict the use of agile practices in the software development context.

Future studies should focus more on the team-level when understanding the use of agile practices and build upon such theories when understanding the dynamics of agile teams in addition to trying to validate the result of the present study using external, or peer-assessed, measurements of skills and agile practices. In future studies, instead of asking questions about the non-technical individual skills, we would suggest items regarding non-technical team-level skills. For example, given that members of the team have enough talent and experience for the kind of work that is conducted, items suggested by Wageman et al. [68] might be useful, like for example, whether everyone in the team has the special skills that are needed for team work. Future studies should also include the macro level of analysis in order to investigate if aspect of team agility could be explained on the organizational level, which there are indications of [52].

Acknowledgements

We would like to thank all the participating companies as well as colleagues helping out in different phases of this research.

References

- [1] Ahmad, S., & Muda, N. A. (2011). An empirical framework design to examine the improvement in software requirements through negotiation. *International Journal on New Computer Architectures and Their Applications*, 1, 599–614.
- [2] Alicke, M. D., Klotz, M. L., Breitenbecher, D. L., Yurak, T. J., & Vredenburg, D. S. (1995). Personal contact, individuation, and the better-than-average effect. *Journal of personality and social psychology*, 68, 804.
- [3] Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological review*, 89, 369.
- [4] Argyris, C. (2000). Double-loop learning. *Wiley Encyclopedia of Management*, .
- [5] Bender, L. L., Walia, G. S., Fagerholm, F., Pagels, M., Nygard, K. E., & Münch, J. (2014). Measurement of the non-technical skills of software professionals: An empirical investigation. In *International Conference on Software Engineering and Knowledge Engineering (SEKE)* (pp. 478–483).
- [6] Booth, C., & Harmer, M. (1994). Agile manufacturing concepts and opportunities in ceramics. *Ceramic Transactions*, 50, 67–76.
- [7] Chamberlain, S., Sharp, H., & Maiden, N. (2006). Towards a framework for integrating agile development and user-centred design. In *Extreme programming and agile processes in software engineering* (pp. 143–153). Springer.
- [8] Chen, G., Gully, S. M., & Eden, D. (2001). Validation of a new general self-efficacy scale. *Organizational research methods*, 4, 62–83.
- [9] Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81, 961–971.
- [10] Cockburn, A., & Highsmith, J. (2001). Agile software development: The people factor. *Computer*, 34, 131–133.
- [11] Cohen, J. (1992). Quantitative methods in psychology – A power primer. *Psychological Bulletin*, 112, 155–159.
- [12] Cohn, M. (2004). Situational leadership for agile software development. *Cutter IT Journal*, 17, 16–21.
- [13] Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, 20, 329–354.
- [14] Conboy, K., Coyle, S., Wang, X., & Pikkarainen, M. (2010). People over process: Key people challenges in agile development. *IEEE Software*, 99, 47–57.
- [15] Coughlan, J., Lycett, M., & Macredie, R. D. (2003). Communication issues in requirements elicitation: A content analysis of stakeholder experiences. *Information and Software Technology*, 45, 525–537.
- [16] Cronbach, L. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, 16, 297–334.
- [17] Crowder, J. A., & Friess, S. (2015). *Agile project management: Managing for success*. Cham, Switzerland: Springer.
- [18] Edum-Fotwe, F. T., & McCaffer, R. (2000). Developing project management competency: Perspectives from the construction industry. *International Journal of Project Management*, 18, 111–124.
- [19] Engel, D., Woolley, A. W., Jing, L. X., Chabris, C. F., & Malone, T. W. (2014). Reading the mind in the eyes or reading between the lines? Theory of mind predicts collective intelligence equally well online and face-to-face. *PloS ONE*, 9, 1–16.
- [20] Fabrigar, L. R., & Wegener, D. T. (2012). *Exploratory factor analysis*. Oxford: Oxford University Press.
- [21] Fletcher, G., McGeorge, P., Flin, R. H., Glavin, R. J., & Maran, N. J. (2002). The role of non-technical skills in anaesthesia: A review of current literature. *British Journal of Anaesthesia*, 88, 418–429.
- [22] Fricker, S. (2010). Requirements value chains: Stakeholder management and requirements engineering in software ecosystems. In *International Working Conference on Requirements Engineering: Foundation for Software Quality* (pp. 60–66).
- [23] Gren, L., Torkar, R., & Feldt, R. (2017). Group development and group maturity when building agile teams: A qualitative and quantitative investigation at eight large companies. *The Journal of Systems and Software*, 124, 104–119.
- [24] Grünbacher, P., & Seyff, N. (2005). Requirements negotiation. In A. Aurum, & C. Wohlin (Eds.), *Engineering and Managing Software Requirements* (pp. 143–158). Berlin: Springer.
- [25] Hackman, J. R. (2003). Learning more by crossing levels: Evidence from airplanes, hospitals, and orchestras. *Journal of organizational behavior*, 24, 905–922.
- [26] Hellmann, T. D., Sharma, A., Ferreira, J., & Maurer, F. (2012). Agile testing: Past, present, and future – Charting a systematic map of testing in agile software development. In *Agile Conference (AGILE), 2012* (pp. 55–63). IEEE.
- [27] Hogg, M. A., & Williams, K. D. (2000). From I to we: Social identity and the collective self. *Group Dynamics: Theory, Research, and Practice*, 4, 81.
- [28] Holck, J., & Jørgensen, N. (2003). Continuous integration and quality assurance: A case study of two open source projects. *Australasian Journal of Information Systems*, 11, 40–53.
- [29] Hudson, N. W., & Fraley, R. C. (2015). Volitional personality trait change: Can people choose to change their personality traits? *Journal of personality and social psychology*, 109, 490.
- [30] Inayat, I., & Salim, S. S. (2015). A framework to study requirements-driven collaboration among agile teams: Findings from two case studies. *Computers in Human Behavior*, 51, 1367–1379.
- [31] Janis, I. L., & Mann, L. (1977). *Decision making: A psychological analysis of conflict, choice, and commitment*. New York: The Free Press.
- [32] Kelle, E. V., Wijst, P. V. D., Visser, J., Plaat, A., & Group, S. I. (2015). An empirical study into social success factors for agile software development. In *International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)* (pp. 6–9).
- [33] King, J. L., & Kraemer, K. L. (1984). Evolution and organizational information systems: An assessment of nolan’s stage model. *Communications of the ACM*, 27, 466–475.
- [34] Klein, K. J., & Kozlowski, S. W. (2000). From micro to meso: Critical steps in conceptualizing and conducting multilevel research. *Organizational research methods*, 3, 211–236.

- [35] Kovitz, B. (2003). Hidden skills that support phased and agile requirements engineering. *Requirements Engineering Journal*, 8, 135–141.
- [36] Laanti, M., Similä, J., & Abrahamsson, P. (2013). Definitions of agile software development and agility. In *Systems, Software and Services Process Improvement* (pp. 247–258). Springer.
- [37] Lalsing, V., Kishnah, S., & Pudaruth, S. (2012). People factors in agile software development and project management. *International Journal of Software Engineering & Applications*, 3, 117–137.
- [38] Larman, C., & Basili, V. R. (2003). Iterative and incremental development: A brief history. *IEEE Computer*, (pp. 47–56).
- [39] Lee, S. M., & Lee, C. K. (2006). IT managers' requisite skills. *Communications of the ACM*, 49, 111–114.
- [40] Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L., & Zelkowitz, M. (2002). Empirical findings in agile methods. In *Extreme Programming and Agile Methods* (pp. 197–207). Springer.
- [41] Liu, L., Maurer, F., & Erdogmus, H. (2005). An environment for collaborative iteration planning. In *Proceedings of the Agile Development Conference* (pp. 80–89). IEEE.
- [42] Maham, M. (2008). Planning and facilitating release retrospectives. In *Agile Conference (AGILE)* (pp. 176–180). IEEE.
- [43] Marquardt, D. W. (1970). Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. *Technometrics*, 12, 591–612.
- [44] McHugh, O., Conboy, K., & Lang, M. (2012). Agile practices: The impact on trust in software project teams. *IEEE Software*, 29, 71–76.
- [45] Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software*, 82, 1869–1890.
- [46] Napier, N. P., Keil, M., & Tan, F. B. (2009). IT project managers' construction of successful project management practice: A repertory grid investigation. *Information Systems Journal*, 19, 255–282.
- [47] Nuseibeh, B., & Easterbrook, S. (2000). Requirements Engineering: A Roadmap. In *Proceedings of the Conference on The Future of Software Engineering (ICSE)* (pp. 35–46).
- [48] van Oosterhout, M. (2010). *Business Agility and Information Technology in Service Organizations*. Rotterdam: Erasmus Research Institute of Management.
- [49] Pentland, B. T., & Feldman, M. S. (2005). Organizational routines as a unit of analysis. *Industrial and corporate change*, 14, 793–815.
- [50] Petersen, K., & Wohlin, C. (2010). The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Software Engineering*, 15, 654–693.
- [51] Reynolds, K. J., Turner, J. C., Haslam, S. A., & Ryan, M. K. (2001). The role of personality and group factors in explaining prejudice. *Journal of Experimental Social Psychology*, 37, 427–434.
- [52] Roth, A. (1996). Achieving strategic agility through economies of knowledge. *Strategy & leadership*, 24, 30–36.
- [53] Salvato, C. (2009). Capabilities unveiled: The role of ordinary activities in the evolution of product development processes. *Organization Science*, 20, 384–409.
- [54] Schwaber, K., & Beedle, M. (2002). *Agile software development with scrum*. Upper Saddle River, NJ: Prentice Hall.
- [55] Serrador, P., & Pinto, J. K. (2015). Does agile work? – A quantitative analysis of agile project success. *International Journal of Project Management*, 33, 1040–1051.
- [56] Sharp, H., & Robinson, H. (2004). An ethnographic study of XP practice. *Empirical Software Engineering*, 9, 353–375.
- [57] Shell, G. R. (2001). Bargaining styles and negotiation: The Thomas-Kilmann conflict mode instrument in negotiation Training. *Negotiation Journal*, (pp. 155–174).
- [58] Sheppard, J., & Young, W. (2006). Agility literature review: Classifications, training and testing. *Journal of sports sciences*, 24, 919–932.
- [59] Sillitti, A., & Succi, G. (2005). Requirements engineering for agile methods. In A. Aurum, & C. Wohlin (Eds.), *Engineering and Managing Software Requirements* (pp. 309–326). Berlin: Springer.
- [60] Skill. (n.d.). Oxford dictionaries – english. <https://en.oxforddictionaries.com/definition/skill>.
- [61] Snyder, M. (1984). When belief creates reality. *Advances in experimental social psychology*, 18, 247–305.
- [62] So, C. (2010). *Making software teams effective: How agile practices lead to project success through teamwork mechanisms*. Frankfurt am Main: Peter Lang.
- [63] So, C., & Scholl, W. (2009). Perceptive agile measurement: New instruments for quantitative studies in the pursuit of the social-psychological effect of agile practices. In *Agile Processes in Software Engineering and Extreme Programming* (pp. 83–93). Springer.
- [64] Stray, V. G., Moe, N. B., & Aurum, A. (2012). Investigating daily team meetings in agile software projects. In *38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 274–281). IEEE.
- [65] Tanner, M., & von Wilingh, U. (2014). Factors leading to the success and failure of agile projects implemented in traditionally waterfall environments. In *Management, Knowledge and Learning International Conference* (pp. 693–701).
- [66] Turley, R. T., & Bieman, J. M. (1994). Identifying essential competencies of software engineers. In *ACM Conference on Computer Science* (pp. 271–278).
- [67] Viller, S., & Sommerville, I. (1999). Social analysis in the requirements engineering process: From ethnography to method. In *IEEE International Symposium on Requirements Engineering* (pp. 6–13). IEEE.
- [68] Wageman, R., Hackman, J. R., & Lehman, E. (2005). Team diagnostic survey: Development of an instrument. *The Journal of Applied Behavioral Science*, 41, 373–398.
- [69] Wang, X., Conboy, K., & Pikkarainen, M. (2012). Assimilation of agile practices in use. *Information Systems Journal*, 22, 435–455.
- [70] Williams, L. (2012). What agile teams think of agile principles. *Communications of the ACM*, 55, 71–76.
- [71] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2000). *Experimentation in Software*

- Engineering: An Introduction*. Norwell, MA, USA: Kluwer Academic Publishers.
- [72] Woolley, A. W., Chabris, C. F., Pentland, A., Hashmi, N., & Malone, T. W. (2010). Evidence for a collective intelligence factor in the performance of human groups. *Science*, *330*, 686–688.