Journal of Software Engineering Research and Development, 2019, 14:17, doi: 10.5753/jserd.2026.6280

The Development and Validation of a Questionnaire to Assess Agility in Software Engineering (QASE)

Vincent Tchikanmou [Europa-Universität Flensburg | vincent-auriol.tchikanmou@promovierende.uni-flensburg.de]
Tabea Scheel [Europa-Universität Flensburg | tabea.scheel@uni-flensburg.de]

Abstract

The concept of agility is increasingly becoming the focus of empirical research. However, what agility is and how it can best be conceptualized and measured remain controversial, even among researchers. This lack of consensus is reflected in the literature in the divergent approaches used to operationalize the concept in research, which limits the comparability of empirical results and impedes the development of a cohesive body of knowledge in the field. Against this background, the present article addresses the development of a valid questionnaire to assess agility in software engineering (QASE) that can be consistently used across various contexts. For this purpose, the commonalities among pioneer agile methods were examined by analyzing their underlying techniques, resulting in a second-order measurement model comprising 30 factors and 90 items. The QASE was evaluated using confirmatory factor analysis in a first study ($N = 418$), then refined and re-evaluated in a second study ($N = 152$), thereby demonstrating its factor structure, validity, and reliability. With the QASE, we provide the research community with a theoretically sound and empirically valid measurement instrument that facilitates hypothesis testing, supports theory development, and enables more reliable conclusions about the mechanisms underlying the positive outcomes associated with agility, thereby contributing to a more coherent and cumulative body of knowledge in Agile Software Development.

Keywords: *Agile, Agile Measurement Scale, Assessment of Agility, Agile Questionnaire, Measurement of Agility, Psychometric*

1 Introduction

For more than two decades, the concept of agility has continuously dominated the management discourse and is becoming increasingly popular (Cico et al., 2021; Gnizy, 2025; Joiner, 2019). What first went mainstream with the crystallization of the Agile Manifesto in the early 2000s and the rise of Agile Software Development has evolved into a widespread phenomenon, leading to the emergence of a multitude of so-called agile neologisms (Madsen, 2020). Today, it is no longer just about Agile Software Development, but also about agile leadership (Akkaya et al., 2022), agile work organization (Greineder et al., 2020), agile organization (Brunet et al., 2021), agile HR (Moh'd et al., 2024), agile governance (De O Luna & Marinho, 2023), or agile innovation (Holden et al., 2021), to name but a few (Madsen, 2020). The latest frontier is "agile everything", defined as a "... solution[] to nearly any problem - whether at work or in life" (Fewell, 2015, p. 1). This broadening of the concept's scope is associated with inconsistencies in its use, which confuses the scientific connotation of the term (Knutsen et al., 2024; Ozkan & Gok, 2022; Wendler, 2013). As a result, despite the extensive literature on agility, its strong reputation among practitioners, and its growing prevalence across various fields, the question of what constitutes agility remains controversial within the academic research community (Baham & Hirschheim, 2021; Conboy, 2009; Laanti et al., 2013; Madsen, 2020; Magistretti & Trabucchi, 2025; Pinho et al., 2022).

In one of the earliest and most comprehensive systematic literature reviews on agile software development — one of the most widely acknowledged and established agile concepts in practice and even considered a sub-discipline of software engineering (Hoda et al., 2018) —, Dybå and Dingsøyr (2008) examined 1,996 articles on agility and found that a mere 36 were empirical studies. Moreover, even among these 36 articles, significant shortcomings were identified in the reporting of research methods and limitations in terms of bias, validity, and reliability (Dybå & Dingsøyr, 2008) - a criticism confirmed and reiterated by subsequent research (e.g., Abrahamsson et al., 2009; Pinho et al., 2022; Tripp et al., 2018). In their work on the tenth anniversary of the Manifesto for Agile Software Development, Dingsøyr et al. (2012) conclude that even after a decade, too little is known about agile methods, as our knowledge of agility consists largely of experience reports, and its appeal is based more on intuition than on empirically sound evidence of its effectiveness.

Over the past decade, numerous studies have been conducted to address this gap, providing empirical support for the benefits of agile approaches (e.g., Bergmann, 2018; De Barros et al., 2024; Haskin, 2022; Karekar, 2024; Lalić et al., 2022; Recker et al., 2017; Serrador & Pinto, 2015; Tripp et al., 2016; Umer et al., 2021). However, despite the valuable insights gained from empirical research on agility, the field suffers from inconsistencies in both the conceptualization and operationalization of the concept, which hinders cohesive research progress (Knutsen et al., 2024). Indeed, what constitutes agility or agile software development has not been theoretically established and represents a long-standing and widely discussed critical gap in the agile literature (Abrahamsson et al., 2009; Baham & Hirschheim, 2021; Conboy, 2009; Knutsen et al., 2024; Laanti et al., 2013; Madsen, 2020; Magistretti & Trabucchi,

2025; Pinho et al., 2022; Silva-Martinez et al., 2024; Tripp et al., 2018). Researchers have therefore studied agility using different definitions, conceptualizations, and operationalizations, resulting in a fragmented body of knowledge (Knutsen et al., 2024). For instance, while Serrador and Pinto (2015) report a positive correlation between agile methods and project success, they define and operationalize agile methods as "the amount of planning effort during the planning and execution phases" (Serrador & Pinto, 2015, p. 1045). Conversely, Rathor et al. (2023) assessed software development agility as the ability to sense, respond, and learn. Other authors refer to various combinations of agile practices (e.g., Memeti et al., 2021; Rietze & Zacher, 2022; Tripp et al., 2016). Still, other empirical studies, as criticized by Tripp et al. (2018), do not measure agility at all but merely state that teams claim to work in an agile manner. As articulated by Baham and Hirschheim (2021, p. 106), "researchers are challenged with attempting to convince readers that the phenomenon that they are studying is "agile" or constantly providing evidence of its differences from waterfall methods without a universally accepted way to report and evaluate such claims".

This lack of consensus on what constitutes agility has not only meant that our knowledge of the concept has remained diffuse for more than two decades (Drechsler & Ahlemann, 2015; Knutsen et al., 2024; Madsen, 2020), but also that, as a result of this diffusion, the term has become susceptible to losing its content and thus its significance through dilution and proliferation of the concept (Fowler, 2006; Thomas, 2014).

In a 2022 technical report, Telemaco et al. (2022) summarize 60 different agile measurement tools published between 2006 and 2020. This does not even include individual by-product measurement tools such as those by Recker et al. (2017), Serrador and Pinto (2015), Tripp et al. (2016), or Bergmann (2018). Even in the same year as the work of Telemaco et al. (2022) was presented, more instruments have been proposed (e.g., Cohard & Messeghem, 2022; Eilers et al., 2022; Petermann & Zacher, 2022; Tuncel et al., 2022). Given the multiplicity of measurement instruments, compounded by inconsistency in the conceptualization of the agile construct, researchers have called "for a systematic review of existing concepts and a psychometric validation study to develop a valid measurement concept that can be used consistently across different studies" (Rietze & Zacher, 2022, p. 19; see also Chronis & Gren, 2016; Junker et al., 2021). Against this background, this research aims to uncover the theoretical core of Agile Software Development — one of the most widely recognized and established agile concepts in practice — and, based on this, to develop a valid measurement instrument to assess agility, thereby reconciling the literature. To achieve this goal, the remainder of this paper is organized as follows. In the next section, we introduce the concept of Agile Software Development and its underlying structure, followed by an overview of previous conceptualization approaches and their respective limitations. In the third section, we present a new approach based on the process underlying the development of the Agile Manifesto and derive from it the theoretical core of Agile Software Development, comprising the central aspects common to all generally accepted agile software development methods (Baham & Hirschheim, 2021; Hohl et al., 2018; Strode, 2005). Using methods from psychometrics, which has a long tradition in the development of measurement instruments, the conceptualized construct is operationalized and statistically validated, and the results are then discussed.

By uncovering the theoretical core of Agile Software Development, this study contributes to a deeper understanding of the concept and provides the research community with a theoretically sound and valid measurement instrument to assess agility in software engineering, thereby laying the foundation for more cohesive research in the field and a cumulative body of knowledge.

2 Theoretical Background

2.1 The Agile Software Development Construct

There are a variety of definitions for the term agility (Laanti et al., 2013), ranging from the "ability to create and respond to change and deal with, and ultimately succeed in, an uncertain and turbulent environment" (Agile Alliance, 2015, para. 1) through a more structural, business-centric description as "dynamic, context-specific, aggressively change-embracing, and growth-oriented" (Goldman et al., 1995, p. 42), to a more systemic perspective in which agility is defined as the "successful exploration of competitive bases (speed, flexibility, innovation proactivity, quality and profitability) through the integration of reconfigurable resources and best practices" (Yusuf et al., 1999, p. 37). This plurality can be explained either by the different facets of the concept, all of which are consequently necessary to grasp it as a whole, or by an imprecise definition of the construct, which gives rise to a paradox of the heap (Burgess, 1990). Etymological knowledge of a concept can provide clues about how to deal with such polysemies (Bridgman, 1950). In the case of agility, however, there is no consensus on its origin (Madsen, 2020; Whiteley et al., 2021) and thus no basis for determining whether one, several, and which definitions are necessary to capture this construct. Some authors explain this definitional variety through the historical evolution of the term, which is deeply rooted in both Lean Management and Agile Manufacturing (Jin‐Hai et al., 2003; Pinho et al., 2022). While Lean Management—originating from the Toyota Production System—provided principles for improving efficiency and eliminating waste, Agile Manufacturing emphasized responsiveness, flexibility, and adaptation to dynamic environments. Both streams provided important conceptual

foundations for the emergence of agile software development, in which these ideas were adapted to address the specific challenges of software development, including rapidly changing requirements, uncertainty, and the need for continuous customer feedback. Consequently, modern definitions often oscillate between these Lean-inspired systemic efficiencies and the mindset-oriented flexibility popularized by the Agile Manifesto (Gunasekaran et al., 2019). Indeed, the crystallization of the "Manifesto for Agile Software Development" (Beck et al., 2001b) in the early 2000s, which led to the institutionalization of agility in so-called agile values and agile principles, is regarded as the constitutionalization of the agile movement (Cohen et al., 2004; Kiv et al., 2018). Since then, these have been regarded as the conceptual foundation of agility (Highsmith & Cockburn, 2001; Laanti et al., 2013).

The Manifesto for Agile Software Development introduces four values (i.e., Individuals and interactions over processes and tools, Working software over comprehensive documentation, Customer collaboration over contract negotiation, Responding to change over following a plan) (Beck et al., 2001b) and twelve principles of agility (e.g., Working software is the primary measure of progress, Continuous attention to technical excellence and good design enhances agility, The best architectures, requirements, and designs emerge from self-organizing teams) (Beck et al., 2001c). The Manifesto for Agile Software Development, however, provides limited guidance on how these values and principles should be interpreted and how they relate to one another. For instance, it is unclear whether adhering to the values alone is sufficient to be considered agile, or whether a combination with the principles is necessary. In the current literature, these are interpreted in a hierarchical means-purpose relationship, wherein the principles represent a concretization of the values (e.g., Ariza et al., 2020; Fuchs & Golenhofen, 2019). In this context, Kiv et al. (2017) and Mortensen et al. (2007) independently assigned the 12 agile principles to the four agile values. However, their findings differ substantially in both numbers and content. This illustrates the interpretive latitude inherent in the agile values and principles, which, like the concept of agility itself, allows for considerable individual interpretation. Conboy and Fitzgerald (2004) therefore consider them unsuitable as a basis for scientific research.

In a comprehensive article, Atzberger et al. (2020) address the topic of ambiguity and taxonomy in the context of agile software development. They posit that the ambiguity inherent in agile values is a necessary quality, enabling adaptability to different contexts. Software projects go through different development life cycles depending on their type and therefore require different development processes, which in turn require a development methodology adapted to the specific life cycle (Acuña et al., 1999; Atzberger et al., 2020; Cockburn, 2002; Itzik & Roy, 2023). Agile software development is, therefore, a philosophy and, as such, is based on values that can only be implemented in practice through agile methods, which serve as context-dependent further concretizations of those values (Atzberger et al., 2020). In this context, Highsmith and Cockburn (2001), two authors of the Manifesto for Agile Software Development, posit that: "What is new about agile methods is not the practices they use, but their recognition of people as the primary drivers of project success, coupled with an intense focus on effectiveness and maneuverability. This yields a new combination of values and principles that define an agile world view." (Highsmith & Cockburn, 2001, p. 122). The agile values and principles are therefore considered the hallmark and pivot of agility, and agile methods those working methods that share the agile values and/or principles and may therefore also be regarded as agile (Cohen et al., 2004).

Over the years, new agile methods have emerged, and today there is disagreement about how many agile methods there are or which methods actually share the agile values and/or principles and thus belong to the agile family (Abrahamsson et al., 2002; Strode, 2005). Even though the agile values and principles define the foundation of all agile methods, it should be noted that they were originally derived from a group of software development methods that inspired the authors of the Manifesto for Agile Software Development to formulate these values (Abrahamsson et al., 2003; Beck et al., 2001a; Cohen et al., 2004; Hohl et al., 2018). The agile values and/or principles can thus be understood as the common denominator underlying these methods. As explicitly mentioned in the epilogue of the Manifesto for Agile Software Development (Beck et al., 2001a), these include **Extreme Programming** (XP; Beck, 1999), **Scrum** (Schwaber & Beedle, 2001), **Dynamic Systems Development Method** (DSDM; Stapleton, 1997), **Adaptive Software Development** (ASD; Highsmith, 2000), **Crystal Family of Methodologies** (CFM; Cockburn, 2000), **Feature-Driven Development** (FDD; Palmer & Felsing, 2002), and **Pragmatic Programming** (PP; Hunt & Thomas, 1999). Based on interviews with the authors of the Manifesto for Agile Software Development, Hohl et al. (2018) list ten agile methods, supplementing the previous list with three additional methods: **Test-Driven Development** (TDD; Beck, 2003), **Modeling Languages** (ML; Ambler, 2002; Mellor & Balcer, 2002), and **Design-Driven Development Refactoring** (3DR; Fowler et al., 1999).

Although all these methods share a common reference to agile values and/or principles, they differ in terms of the aspects of the life cycle they cover, the development roles they envisage, and the activities they define for each role (Abrahamsson et al., 2002; Cockburn, 2002). These contextual properties account for the existence of numerous agile methods, the distinctions between them, and their suitability or unsuitability for certain types of projects (Abrahamsson et al., 2003; Strode, 2005). Consequently, some authors have proposed integrating several agile methods into a comprehensive methodology to bridge the gaps between the various methods and enable their

application to a broad range of software projects (Darwish, 2014; Mushtaq & Qureshi, 2012; Sharma & Wadhwa, 2015). However, the distinction between the terms process, method, and methodology is a matter of contention in software development, which is why they are often used interchangeably (Acuña et al., 1999). For instance, although the Manifesto for Agile Software Development explicitly uses the term "methodologies" to refer to certain approaches, other authors use the terms "method", "practice" (Hohl et al., 2018), "framework", or "process model" (Atzberger et al., 2020). Consequently, the extent to which the agile approaches mentioned in the epilogue of the Manifesto for Agile Software Development are "methodologies" that fully embody all agile values and principles, or whether their combination is required to be considered agile, is not fully established. For instance, Visconti and Cook (2004) analyze two of the most frequently used agile approaches, Extreme Programming and Scrum. They conclude that neither approach individually covers all 12 agile principles and suggest combining the two. Consequently, it remains unclear which agile practices contribute to the fulfillment of which values or principles and thus which methods are fully agile (Conboy & Fitzgerald, 2004). In the present study, we use the term "agile methods" in a broader, functional sense, as a general reference to all agile software development approaches (including frameworks such as Scrum, methods such as Extreme Programming, and hybrid approaches) as a means to implement agility in practice.

To summarize, the concept of agility is conceptually elusive. Its philosophical foundation consists of the agile values and principles, which are implemented in practice through agile methods and their respective techniques. However, it is not established which agile techniques support specific values or principles, and thus which methods can be considered fully agile (Conboy & Fitzgerald, 2004).

2.2 Existing Conceptualizations of Agility and Their Limitations

Based on the preceding conceptual analysis, the degree of agility of a project can be defined as the extent to which agile values and/or principles are fulfilled and can therefore be conceptualized and operationalized on this basis (e.g., Haase et al., 2017). However, irrespective of the unknown relationship between agile values and principles, this approach proves unsuitable for effective operationalization of agility due to their interpretive latitude, as demonstrated by the results of Haase (2017). Each interpretation results in a distinct measurement instrument that not only differs in content but also yields disparate outcomes (Chronis & Gren, 2016; Jalali et al., 2014).

Conceptualization at the level of the method in relation to its concrete design represents another possibility. Michaelides et al. (2010), for instance, pursue this approach. However, it entails several challenges and limitations. Firstly, it is uncertain whether and which agile methods encompass all values and/or principles. Secondly, a measurement tool that covers only a single agile method can only be useful for projects that use this method. An international survey conducted by Kuhrmann et al. (2021) revealed that over 85% of the 556 respondents employed hybrid methods that combine several agile or non-agile methods. This suggests that agile methods are often adapted to the requirements of the project in practice. Consequently, relying on a single agile method to conceptualize and operationalize agility is inadequate and impractical, as it may fail to capture all aspects of the agile construct and limit the applicability of the resulting measurement instrument in practice and the comparability of research results.

Other authors (e.g., Junker et al., 2022; So & Scholl, 2009; Tripp et al., 2016) draw on agile practices from disparate methods to conceptualize agility. While this procedure is among the most common approaches in the literature, it is constrained by the lack of an established understanding of which agile practices contribute to the realization of which agile values and principles, thereby making it difficult to ensure completeness. Baham and Hirschheim (2021) raise another concern regarding this approach, arguing that the techniques within a method constitute "a 'system of practices', reinforcing and tightly coupled" (Baham & Hirschheim, 2021, p. 120). A measurement instrument that is limited to specific practices therefore cannot account for this reciprocal effect (Baham & Hirschheim, 2021). Moreover, practices are only one of many components of a software method, along with processes, roles, or tools, which therefore likewise need to be considered to capture the concept of agility as a whole (Cockburn, 2002). Consequently, the comparability and integration of research findings are constrained, as meaningful comparisons are only possible between studies that investigate the same set of agile practices.

Beyond the conceptualizations of agility discussed above, other studies conceptualize agility based on a definition (e.g., Geiger, 2020; Meyer et al., 2021; Yauch, 2011). However, given the lack of consensus on a definition of agility and the inherent interpretive latitude of the concept, such conceptualizations are not falsifiable, and their accuracy cannot be verified due to a lack of references. As Madsen (2020) points out, the concept of agility is not only an elusive construct but also exhibits a high degree of "interpretative viability," a trait associated with concepts that can be understood and interpreted differently by different actors. Consequently, the fundamental prerequisite for a valid definition-based conceptualization has not been met (Sjøberg & Bergersen, 2023). This issue is particularly evident in the discussion surrounding the relevance of instruments commonly referred to as Agile Maturity Models (Corrêa Rodrigues & Mantovani Fontana, 2019; Leppänen, 2013; Schweigert et al., 2013; Tuncel et al., 2020; Vasylieva et al., 2023). These include, for instance, 57 of the 60 agile measurement instruments summarized by Telemaco et al.

(2022). Indeed, despite their prevalence in the literature, their validity is highly controversial among researchers, as the underlying theoretical background, if presented at all, consists of subjective reasoning without a solid theoretical foundation and empirical validation of the proposed models (Henriques & Tanner, 2017). Gren et al. (2015, p. 25) conclude on the basis of their validation study of an agile maturity model that "researchers cannot correlate quantitative agile maturity measurements to other variables in Software Engineering research and be confident that the results are correct. Practitioners cannot either use these tools to guide their journey towards agility." Chronis and Gren (2016) reach a similar conclusion when validating three other agile maturity models.

The interpretive latitude inherent in the concept of agility, its values, and principles, along with the indeterminacy regarding which specific agile methods embody them, poses challenges to the effective operationalization of this construct. However, the conceptual relationship between agile values/principles and methods is known. The values and principles are regarded as the essence of the concept of agility and represent the commonality of all agile methods. Consequently, a suitable and effective approach to conceptualizing agility is to examine the commonalities of all agile methods. However, given the interpretive latitude inherent in agile values and principles, the commonalities underlying agile methods should be examined at a level of abstraction below that of the values and principles. Therefore, identifying the commonalities of all agile methods based on their descriptions is better suited to this approach. As it is not generally established which working methods should be considered agile and which should not, the ten original agile methods, from which the authors of the Manifesto for Agile Software Development derived the agile values and principles as the common characteristics of all agile methods, are suitable for this analysis. Accordingly, we conceptualize agility, akin to the agile values and principles, as common characteristics of the ten original agile methods previously mentioned. We restricted our analysis to these 10 foundational agile methods to remain close to the rationale underlying the development of the Agile Manifesto and the identification of agile values and principles as common characteristics of all agile methods and as the foundation of agility. By focusing on these historically influential methods, the present study aims to derive a theoretically grounded conceptualization of agility that reflects the foundational ideas of agile software development.

3 Research Model Specification

3.1 Conceptualization of Agility

To identify the commonalities among the agile methods mentioned above, we first analyzed each method individually. This entailed decomposing them into their components and categorizing them according to a uniform scheme. In the second step, we compared the identified components across the methods category by category and determined commonalities.

We analyzed and categorized agile methods following Cockburn's "Scope of a Methodology" (2002). Abrahamsson et al. (2002) provided a categorization of several agile methods based on the same scheme, which we adopted and extended by including two additional methods relevant to our study: **Test-Driven Development** (Beck, 2003) and **Design-Driven Development Refactoring** (Fowler et al., 1999). We then systematically analyzed the descriptions of each method and extracted the elements for each category. This first analysis yields a total of 187 elements across methods, categorized as process (52), role (46), and practice (89). Among the process and role elements, we considered not only the different phases and roles, but also their characteristics and specificities, such as process outcomes or responsibility for role selection. Tools such as configuration management or printing whiteboards were counted among the practices.

We compared the results of this first analysis step with those reported by Strode (2005) to assess their plausibility. Strode (2005), however, refers only to five of the ten methods considered in the present study, without taking the role into account. Nevertheless, the number of elements identified in our study with the same number of methods and without considering the role is even twice as high as that reported by Strode (2005) and encompasses them all. The results of this initial analysis are shown in Figure 1.

Based on the results of the preceding classification, the methods were compared in a second step. This entailed identifying similarities and complementarities among the elements within each category (process, roles, and practices), beginning with the process category.

In the content analysis of the process elements, we assigned 26 of the 52 identified process elements to the practice category because they do not describe a process phase and its sequence, but rather the process results and how they should be generated. For instance, "Time-boxed" is listed as a process characteristic in the Dynamic Systems Development method but can also be considered a project planning technique (Miranda, 2011). The remaining 26 process elements originate from seven of the ten agile methods analyzed, as no information on process design was available for Test-Driven Development (Beck, 2003), Agile Modeling (Ambler, 2002; Mellor & Balcer, 2002), and Design-Driven Development Refactoring (Fowler et al., 1999). These seven methods all describe a process in which the development phase runs iteratively and incrementally, while the remaining phases are sequential. Therefore, the seven process elements were consolidated into an element, "***Iterative and Incremental Development Phase,"*** as a common feature. This result is consistent with previous studies that consider iterative and incremental development processes one of the main characteristics of agile methods (e.g., Baham & Hirschheim, 2021; Miller, 2001; Strode, 2005). The remaining 19 process elements describe process

steps that are specific to the individual methods investigated. These include, for example, Exploration Phase (XP), Project Initiation (ASD), or Business Study (DSDM).

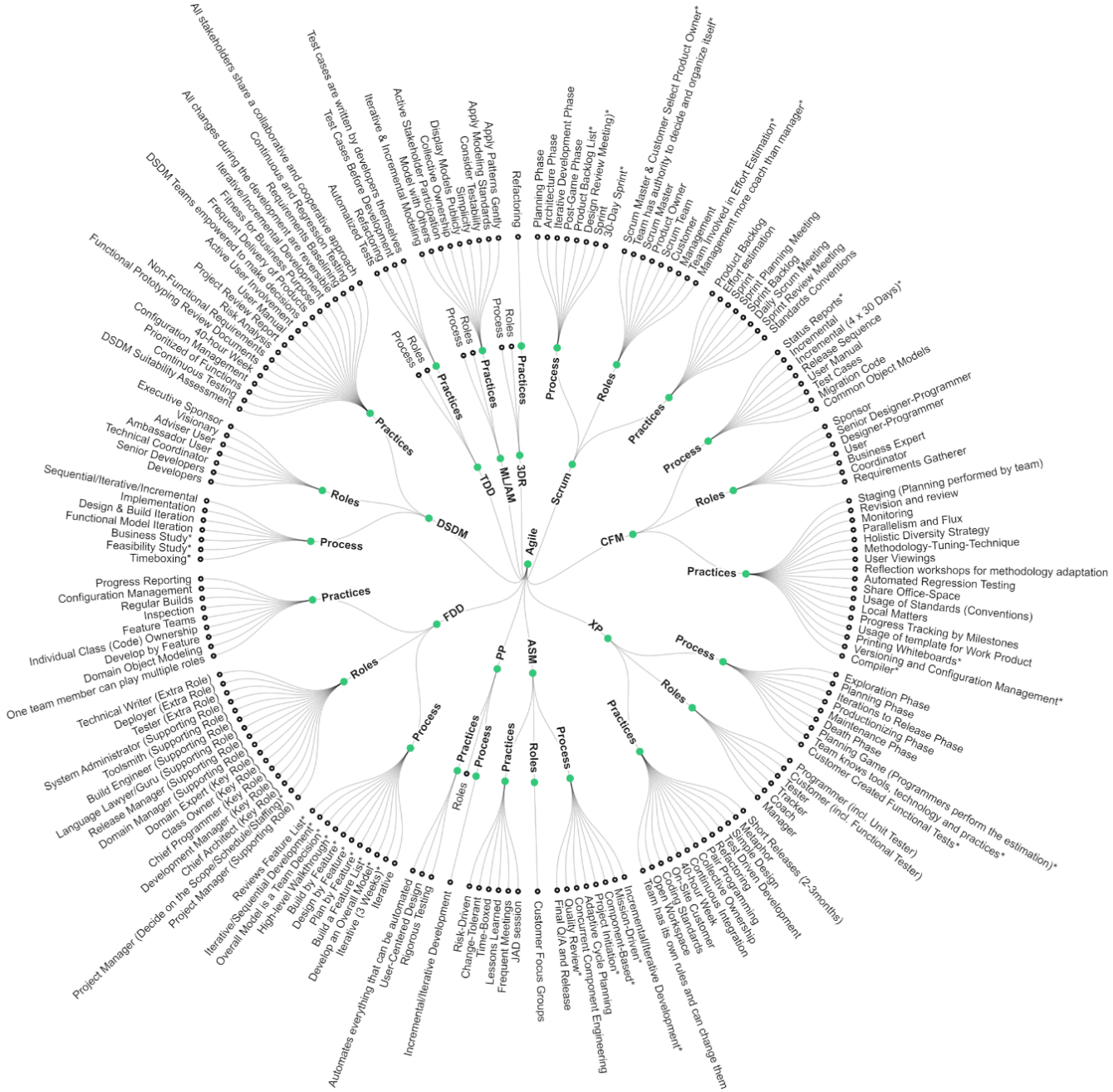

Figure 1: Agile Methods with Their Respective Practices, Roles, and Process Descriptions
Note: * indicates techniques that could not be clearly assigned to a process, role, or practice. CFM _ Crystal Family of Methodologies; XP _ Extreme Programming; ASD _ Adaptive Software Development; PP _ Pragmatic Programming; DSDM _ Dynamic Systems Development Method; TDD _ Test-Driven Development; ML _ Modeling Languages; AM _ Agile Modeling; 3DR _ Design-Driven Development Refactoring; FDD _ Feature-Driven Development.

Following the same procedure, we analyzed the role elements and reclassified five of the original 46 role elements as practices. The remaining 41 elements originate from six of the ten analyzed methods, while the four methods Test-Driven Development (Beck, 2003), Agile Modeling (Ambler, 2002; Mellor & Balcer, 2002), Design-Driven Development Refactoring (Fowler et al., 1999), and Pragmatic Programming (Hunt & Thomas, 1999) do not provide information related to role allocation or role characteristics. The descriptions of the six methods indicate that five define only a general developer role without further differentiation among developer roles. In addition, four out of the six methods specify the need for experienced developers. The Adaptive Software

Development method (Highsmith, 2000) does not specify formal role allocation or detailed role definitions within its process. Instead, it identifies the customer, referred to as the Customer Focus Group, as part of the development team. Similarly, all the other methods, except for Feature-Driven Development (Palmer & Felsing, 2002), involve the customer or user as part of the development team. Another common element found in the role descriptions is that of the manager. The manager is described as involved in the development phase. In the Crystal Family of Methodologies (Cockburn, 2000) and the Dynamic Systems Development Method (Stapleton, 1997), this role is referred to as the sponsor. As a result, we identified four characteristics as common features of the team structure of the analyzed agile methods: ***(1) the assignment of several development roles to each team member, (2) the presence of an experienced developer in the team, (3) the customer or user, and (4) management as part of the development team***. For 22 role elements, none of the four characteristics identified above could be assigned, nor could a new characteristic be derived from role elements occurring in at least two different methods. These include, for example, role elements such as Toolsmith (FDD), Tracker (XP), or Visionary (DSDM).

From the previous content analysis of the process and role elements, the original categorization of 52 process, 46 role, and 89 practice elements was revised to 26 process, 41 role, and 120 practice elements. The resulting categorization was further summarized into one process and four role characteristics. These characteristics form the commonalities of the 26 process and 41 role elements. The aggregation of the process and role elements into their identified commonalities is shown in Figure 2.

To identify commonalities among the 120 agile practices, we performed cluster analysis, grouping the same, similar, and complementary agile practices. Agile practices are considered similar when they pursue the same objective but prescribe different ways of achieving it. For instance, while Scrum prescribes a 30-day sprint, XP requires a 2–3-month release. Both practices are therefore grouped as *Short Release Cycles*. Complementarity, on the other hand, refers to practices that address the same aspect but are expressed through different concrete manifestations. For instance, while XP requires a coach, CFM requires a senior designer-programmer, so both practices were aggregated as "Competence Coverage", reflecting the broader characteristic of ensuring expertise within the team.

In this way, we consolidated 97 out of the 120 agile practices into 25 clusters. These include ***(1) Pair Programming/Modeling, (2) Simple Design, (3) Mission-Driven Development, (4) Short Release Cycles, (5) Continuous Project Monitoring, (6) Team-Based Planning, (7) Feature-Driven Development, (8) Shared Product Understanding, (9) Retrospective, (10) Collective Ownership, (11) Function Prioritization, (12) Decentralized Decision-Making, (13) Sustainable Work Pace, (14) Refactoring, (15) Change and Configuration Management, (16) Standards-Based Development, (17) Documentation Commitment, (18) Automation, (19) Change Tolerant, (20) Timeboxing, (21) Regular Meeting, (22) Open Workspace, (23) Review Practice, (24) Test-Driven Development, (25) Continuous Integration.*** Each cluster comprises practices derived from at least two distinct methods, thus at least two practices per cluster. The remaining 23 practices could not be assigned to any of the 25 practice clusters. Examples include Domain Object Modeling (FDD), Teams familiar with tools, technology, and practices (XP), or Project manager making decisions related to scope, school, and staffing (FDD).

Practices and roles complement the process by providing concrete descriptions of how and by whom process steps are to be fulfilled. We therefore examined whether the 19 processes, 22 roles, and 23 practices for which no commonalities had been identified shared cross-category commonalities or whether they could be mapped to one of the 30 commonalities already identified. As a result, 16 of the 23 practices could be allocated to the commonalities previously identified in the process and role analysis.

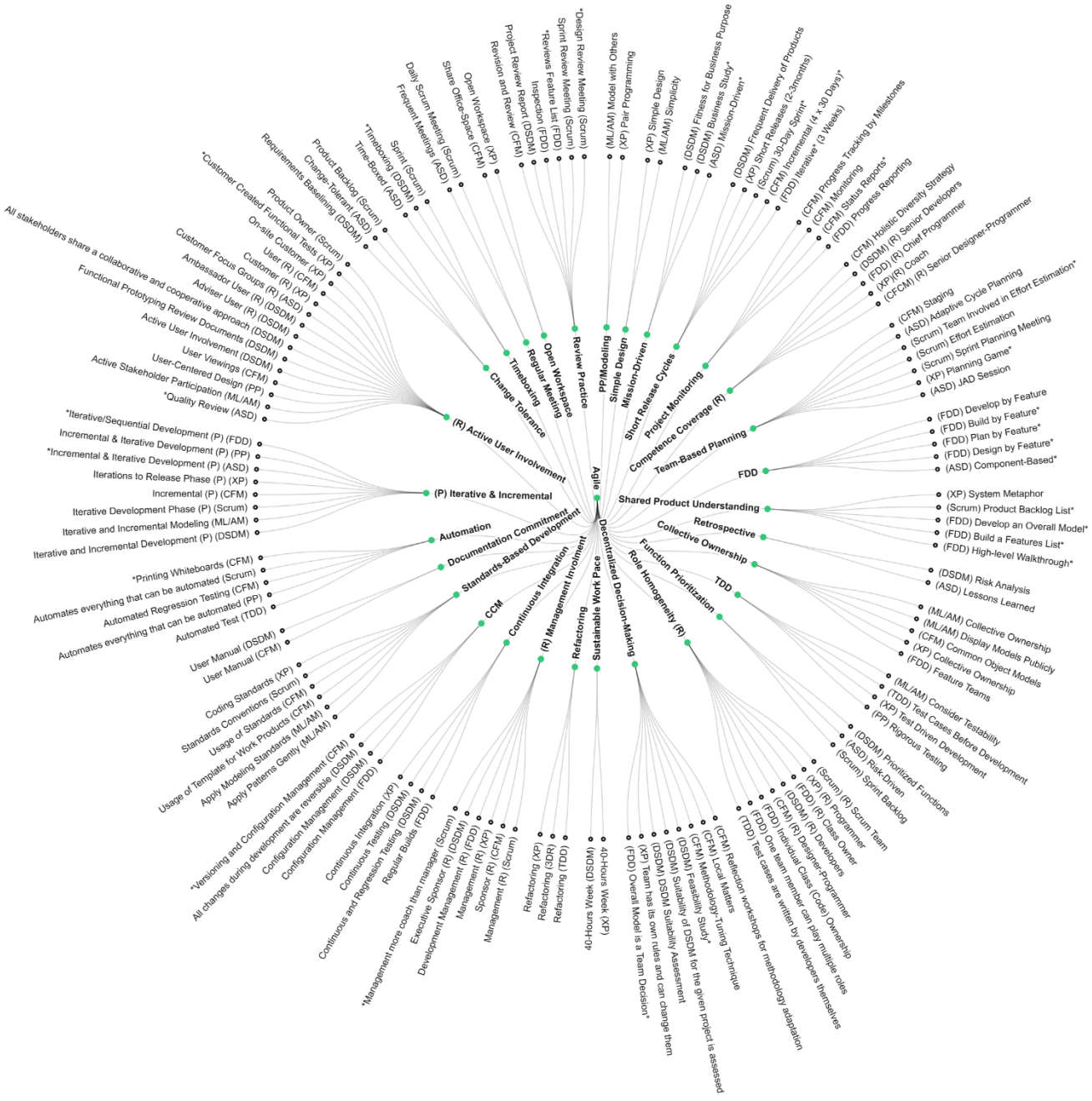

Figure 2: Commonalities of agile methods and their references
Note: The sources of the techniques are presented in parentheses: CFM _ Crystal Family of Methodologies; XP _ Extreme Programming; ASD _ Adaptive Software Development; PP _ Pragmatic Programming; DSDM _ Dynamic Systems Development Method; TDD _ Test-Driven Development; ML _ Modeling Languages; AM _ Agile Modeling; 3DR _ Design-Driven Development Refactoring; FDD _ Feature-Driven Development. CCM _ Change and Configuration Management. The types of techniques are presented in brackets: R _ Role; P _ Process. If not indicated, the technique represents practice.

In summary, 187 elements were initially identified from the analysis of the 10 agile methods examined. The analysis of these 187 elements revealed that 139 exhibited commonalities, meaning that an element from one agile method had an equivalent element in another agile method. The remaining 48 elements could not be associated with any commonality and were specific to a single agile method. A total of 30 commonalities were identified from the 139 elements. This represents the highest level of commonality among the ten agile methods examined, as techniques (processes, roles, and practices) from at least two methods were required to form a commonality. Each commonality therefore represents a distinct dimension of the agile construct. The aggregation of the 139 elements into their respective dimensions is shown in Figure 2.

3.2 Operationalization of Agility

The conceptualization of the latent construct of Agile Software Development yields a 30-factor model in which the respective factors (dimensions) formatively constitute the agile construct. The model's formative character is inherent, as the techniques from which the 30 dimensions arise are not effects of agility but rather contribute to it (Coltman et al., 2008; Diamantopoulos et al., 2008). The identified dimensions, on the other hand, can be specified either formatively or reflectively. However, unlike formative models, reflective models allow for a more accurate estimation of the error component in the measurement model. Therefore, a reflective operationalization of a construct is preferred when the nature of the construct allows for both (Diamantopoulos, 2006). Consequently, we opted for a reflective specification of the respective agile dimensions. This yields a second-order measurement model of the agile construct, as depicted in Figure 3.

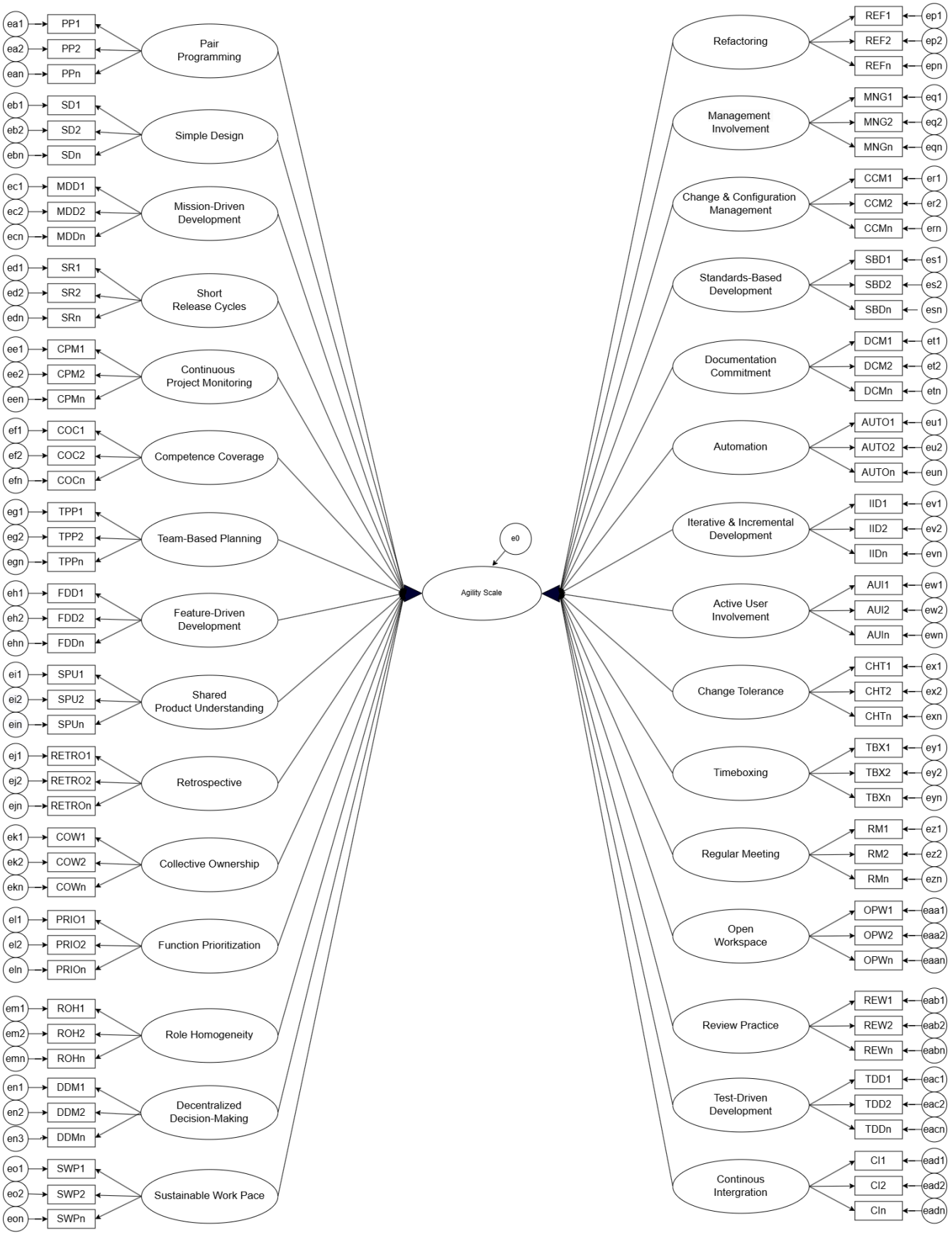

Figure 3: Factor Structure of the Research Model

Note: The ellipses each represent latent variables. Starting with Agility in the middle to the respective 30 agile dimensions. The squares, on the other hand, represent the respective manifest variables (indicators) that are used to measure the latent variables. The circles represent the error term of the variable. Depending on the complexity of the latent variable, a different number of items (from 1 to n) will be required to capture it.

To operationalize the first-order measurement model with reflective indicators, we followed Churchill's (1979) scale development procedure. Accordingly, an initial pool of items was generated for each of the 30 agile dimensions from existing scales (Gren et al., 2015; Looks et al., 2021; Michaelides et al., 2010; So & Scholl, 2009; Tripp et al., 2016; Williams et al., 2004) as well as from self-developed items to capture their key aspects. The resulting item pool was iteratively refined and subjected to two expert reviews. Due to the large number of factors, special care was taken to keep the number of items as small as possible. Consequently, 128 of the initial pool of 211 items were retained for further analysis. The assignment of the items to the respective agile dimensions is presented in Appendix A (see Model 1/M1).

4 Study 1: Measurement Model Evaluation

4.1 Methods

To evaluate the validity of the developed measurement model, we collected data through an online survey using Unipark. Participants were recruited through the authors' professional contacts and software developers in online networks, and were invited to complete the 128-item questionnaire along with eight demographic questions. The 128 items were evaluated using a seven-point Likert scale (*1=Strongly disagree, 2=Disagree, 3=Somewhat disagree, 4=Neither agree nor disagree, 5=Somewhat agree, 6=Agree, and 7=Strongly agree*), supplemented by an *"I don't know"* option (8=*I don't know*).

To reduce the risk of common method variance, participants were informed at the beginning of the survey about the anonymity and confidentiality of data collection. We emphasized that there are no right or wrong answers, that the questions do not refer to their supervisor or the company where they work, and that they are intended only to measure the degree of agility in their project. Furthermore, we designed the survey to provide participants with an assessment of the project's agility upon completion. Respondents were informed about this individual feedback at the beginning of the survey to encourage survey completion and further reduce the risk of common method variance. Prior to data collection and analysis, the study's hypotheses and methods were pre-registered on the Open Science Framework (OSF) (https://osf.io/5epa6). All subsequent analyses were conducted as pre-registered, unless otherwise indicated.

We recruited from March 21, 2022, to August 23, 2022. A total of 2,308 people accepted the invitation. Of these, 1,172 started answering the questionnaire, 686 dropped out, and 448 completed the survey. After data cleaning, 22 records were excluded. The remaining data ($N = 426$) had a high proportion of cases containing missing values (35.7%). The missing values pertain to the questions that were answered with "*I don't know*", since answering all questions was mandatory for the questionnaire to continue. Participants could answer "*I don't know*" if, for example, they did not understand a question, did not want to answer, or lacked sufficient information regarding the usage of the technique in the project. The missing value in the latter case suggests that the technique mentioned is not used in this project, at least from the participant's perspective. This is the case, for example, when a project member responsible only for testing activities needs to answer questions about pair programming. Given the circumstances, missing values were either replaced with the average of existing values or with the value 1 (*Strongly disagree*). In cases where at least two questions were answered within a scale, we imputed the missing values with the average of existing values. In cases where none or only one question was answered within a scale, and the participant stated in the demographic questions that they do not use the related agile technique in their project, we imputed the missing values with the value 1 (*Strongly disagree*). All missing values, except for eight cases, could thus be imputed. This results in a final sample size of $N = 418$ for data analysis.

The 418 participants were drawn from 54 different countries. The largest group was from India (29.2%), followed by Germany (15.3%) and the USA (4.8%). Among the ten original agile methods that contributed to the creation of the Manifesto for Agile Software Development, Scrum was the most commonly used method (54.3%), followed by Feature-Driven Development (7.9%), Agile Modeling (7.7%), and Test-Driven Development (2.6%). Additionally, 14.6% of respondents reported using a combination of at least two of the ten methods, while 6.7% reported not using any of the methods. With respect to work location, 9.8% of respondents indicated that they work exclusively from home, 11.7% reported that they work exclusively in the office, 50.7% stated that they alternate between home and office to varying degrees, and 27.8% did not provide any information. The most common project roles were Programmers (39.2%), Scrum Masters (11.2%), Software Architects (9.3%), Team Leaders (7.9%), Testers (7.2%), and Project Managers (5.3%).

Given that the measurement model was derived from an established theoretical foundation, we conducted a confirmatory factor analysis (CFA) with the cleaned dataset ($N = 418$) using JASP (version 0.16.3.0). While Exploratory Factor Analysis (EFA) is typically used to identify latent structures without a priori assumptions, CFA tests whether empirical data are consistent with a predefined theoretical model. This step is essential for assessing the construct validity and overall model fit of the proposed measurement structure, thereby providing a rigorous psychometric foundation for the subsequent analyses.

The data did not follow a multivariate normal distribution, and the sample size was relatively small ($N = 418$). We therefore conducted the CFA using the robust diagonally weighted least squares (DWLS) estimation method, as recommended in the literature for non-normal data and relatively small samples (Li, 2016; Mîndrilă, 2010;

Rhemtulla et al., 2012). Given the large number of measurement items (128), we conducted a scale purification following the guidelines of Wieland et al. (2017), while considering validity, reliability, and parsimony. Consequently, in addition to the hypothesized model (**Model 1**), we tested two alternative models, each with fewer measurement items per factor than the hypothesized model, but with the same factor structure. The first alternative model (**Model 2**) includes only those items that maximize the internal consistency reliability (McDonald's ω) of each factor. The second alternative model (**Model 3**), on the other hand, includes the three items that contribute most to the internal consistency reliability (McDonald's ω) per factor. Limiting each factor to a maximum of three items in Model 3 was intended to obtain a practically applicable model, thereby reducing the total number of items from 128 to 90. The assignment of the items to the respective models is presented in Appendix A (see Models M1, M2, and M3).In addition to factor structure analysis (CFA), we analyzed the psychometric properties of the QASE to evaluate the scale's statistical strengths and weaknesses. This includes item analysis (factor loadings), reliability analyses (internal consistency, composite reliability), and validity analyses (convergent validity, discriminant validity). To evaluate these properties, we focused in particular on McDonald's ω, Congeneric Reliability (CR), and Average Variance Extracted (AVE). McDonald's ω and CR are measures of the internal consistency of a construct, indicating the extent to which the indicators of a latent construct consistently measure the same underlying concept. In accordance with the prevailing recommendations in psychometric research (Hair et al., 2011; Lance et al., 2006; Nunnally, 1978; Peterson & Kim, 2013), a threshold of CR > 0.7 was adopted as the criterion for acceptable reliability. AVE, in turn, is a measure of convergent validity, indicating the proportion of variance captured by a construct relative to that attributable to measurement error. An AVE value greater than 0.5 is commonly recommended, indicating that the construct explains more than half of the variance in its indicators (Fornell & Larcker, 1981; Hair et al., 2011).

4.2 Results

4.2.1 Factor Structure Analysis

All three models demonstrate good model fit, indicating that the observed data align well with the relationships suggested by the hypothesized models (Hu & Bentler, 1999). However, Model 3 exhibits a superior fit across all indices, while Models 1 and 2 show only slight differences (see Table 1).

Although the fit indicators confirmed the 30-factor structure for all models, Model 1 exhibits a high latent covariance (0.826) between the factors *(29) Test-Driven Development* and *(30) Continuous Integration*. This suggests substantial empirical overlap between the two factors, which may indicate that they are not empirically distinguishable (in this case, an empirical indication of a 29-factor model) or that one or both factors have not been optimally operationalized within the hypothesized 30-factor model.

The techniques of Test-Driven Development and Continuous Integration both refer to getting early feedback on the development steps taken or changes made to a software element. However, in the first case, the focus of the feedback is on the effects of the change on the adapted software element itself. In the second case, the focus of the feedback is on the effects of the change on the entire software, taking into account the dependencies on all other software elements (Amrit & Meijberg, 2018). In practice, both techniques are often combined and automatically executed, jointly referred to as Continuous X or DevOps (Azad & Hyrynsalmi, 2023; Vihovde & Meng, 2024). Consequently, a distinction between the two often becomes blurred in practice. It would therefore be theoretically justifiable to either combine or separately consider both factors. However, combining the two factors into one did not provide sufficient convergent validity (assessed by the average variance extracted (AVE)) of the items. This was also lower than that of the respective factors when considered separately. Moreover, given that the covariance between the two factors in Model 3 is only 0.694, no adjustments were made, and the 30-factor structure was retained.

Table 1: Goodness-of-fit indices of the tested models in Study 1

Model	χ^2	*df*	χ^2/df	*p*	CF	RMSEA	SRMR
Model 1	9956.682	7565	1.316	< .001	.993	.038	.047
Model 2	8929.960	6704	1.332	< .001	.993	.037	.047
Model 3	5039.975	3480	1.448	< .001	.995	.032	.043

Note: For all models, N = 418. CFI – Comparative Fit Index; RMSEA – Root-Mean-Square Error of Approximation; SRMR – Standardized Root-Mean-Square Residual. **Model 1:** hypothesized model (128 items), **Model 2:** model with the highest factor-level McDonald's ω (121 items), **Model 3:** three-item-per-factor model (90 items).

4.2.2 Item, Reliability, and Validity Analysis

The factor loadings and descriptive statistics for all items are presented in Table S1 in the supplementary materials. The items in Models 1 and 2 exhibit comparable factor loadings, except for *(2) Simple Design, (3) Mission-Driven Development*, *(12) Function Prioritization*, *(19) Standards-Based Development*, and *(25) Timeboxing*. These represent the factors for which internal consistency reliability (McDonald's ω) increased after one or more items had been removed. Except for a few items, the factor loadings in Model 3 are higher than in Models 1 and 2. The loadings range from 0.520 to 0.921, thus exceeding the recommended minimum threshold of 0.4 (Hulland, 1999).

All three models exhibit reliable internal consistency with McDonald's ω coefficients between 0.635 and 0.925 (see Table 2). As expected, following item purification,

Model 2 exhibits the highest reliability across the respective factors. However, the largest difference in reliability between Model 2 and Model 3 is 0.06 (see *(30) Continuous Integration*). This is considered acceptable given the advantages of the more parsimonious Model 3. In Model 3, only three factors—*(22) Iterative & Incremental Development*, *(25) Timeboxing*, and *(27) Open Workspace*—showed McDonald's ω values below the recommended threshold of 0.7 (Hair et al., 2011; Lance et al., 2006; Nunnally, 1978; Peterson & Kim, 2013), with coefficients of 0.635, 0.667, and 0.678, respectively. All remaining factors exceeded this threshold. Congeneric reliability (CR) in Model 3 ranges from 0.675 to 0.925. Only factor *(22) Iterative & Incremental Development* fell slightly below the recommended threshold of 0.7 (CR = 0.675) (Hair et al., 2011; Lance et al., 2006; Nunnally, 1978; Peterson & Kim, 2013), whereas all remaining factors exceeded it.

Table 2: Congeneric and construct reliability and convergent validity of the tested factors in Study 1

		McDonald's ω			CR			AVE		
N°	Factor	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3
1	Pair programming/Modeling	.743	.743	.743	.750	.750	.752	.513	.514	.513
2	Simple Design	.830	.835	.835	.875	.865	.865	.637	.681	.681
3	Mission-Driven Development	.822	.852	.852	.886	.897	.897	.613	.745	.745
4	Short Release Cycles	.790	.790	.771	.818	.817	.795	.531	.531	.567
5	Continuous Project Monitoring	.877	.877	.877	.904	.904	.903	.759	.760	.757
6	Competence Coverage	.772	.772	.772	.831	.831	.831	.623	.623	.621
7	Team-Based Planning	.857	.857	.826	.888	.887	.862	.664	.662	.676
8	Feature-Driven Development	.776	.776	.739	.814	.814	.773	.527	.527	.535
9	Shared Product Understanding	.889	.889	.870	.909	.909	.893	.668	.667	.736
10	Retrospective	.913	.913	.895	.934	.934	.918	.740	.740	.789
11	Collective Ownership	.810	.810	.807	.855	.855	.857	.597	.597	.668
12	Function Prioritization	.867	.881	.881	.920	.926	.925	.696	.806	.805
13	Role Homogeneity	.723	.723	.723	.741	.741	.741	.497	.497	.495
14	Decentralized Decision-Making	.839	.839	.786	.871	.871	.816	.578	.578	.601
15	Sustainable Work Pace	.866	.866	.866	.894	.894	.894	.738	.738	.738
16	Refactoring	.849	.849	.847	.908	.908	.908	.712	.712	.768
17	Management Involvement	.811	.811	.800	.859	.859	.847	.605	.605	.651
18	Change and Configuration Management	.879	.879	.829	.901	.901	.849	.604	.603	.652
19	Standards-Based Development	.862	.865	.865	.892	.891	.890	.675	.731	.729
20	Documentation Commitment	.907	.907	.907	.921	.921	.920	.796	.795	.793
21	Automation	.844	.844	.824	.865	.865	.845	.616	.616	.645
22	Iterative & Incremental Development	.659	.659	.635	.705	.705	.675	.378	.378	.410
23	Active User Involvement	.867	.867	.843	.893	.893	.874	.628	.628	.699
24	Change Tolerance	.784	.784	.771	.849	.849	.829	.584	.584	.617
25	Timeboxing	.692	.700	.667	.742	.752	.706	.368	.435	.448
26	Regular Meeting	.874	.874	.852	.912	.912	.890	.675	.675	.729
27	Open Workspace	.681	.681	.678	.758	.758	.751	.450	.451	.507
28	Review Practice	.823	.823	.807	.869	.869	.835	.625	.625	.630
29	Test-Driven Development	.766	.766	.766	.795	.795	.797	.494	.494	.569
30	Continuous Integration	.836	.836	.774	.873	.873	.813	.465	.465	.593

Note: For all factors, $N = 418$. CR _ Congeneric reliability; AVE _ Average Variance Extracted. **Model 1**: hypothesized model (128 items), **Model 2**: model with the highest factor-level McDonald's ω (121 items), **Model 3**: three-item-per-factor model (90 items).

The factors of Model 3 demonstrate greater convergent validity than those of Models 1 and 2 (see Table 2). Except for three factors—*(13) Role Homogeneity*, *(22) Iterative & Incremental Development*, and *(25) Timeboxing*, with AVE values of 0.495, 0.410, and 0.448, respectively—all factors in Model 3 exhibited AVE values above 0.5. This is regarded as an indicator of convergent validity (Fornell & Larcker, 1981; Hair et al., 2011).

To assess the discriminant validity of each factor, we calculated the mean score of the items comprising the respective factor in Model 3 and analyzed the correlation between the resulting factor scores (see Table S2 in the supplementary materials). The factors *(12) Function Prioritization* and *(9) Shared Product Understanding* exhibit the highest bivariate correlation, with a value of 0.603. This is below the defined threshold value of 0.85 and thus represents an initial indication of the discriminant validity of the respective factor (Cheung et al., 2024). Fornell and Larcker (1981) additionally recommend comparing the AVE of each factor with its squared bivariate correlation with all other factors. The squared correlations should be lower than the corresponding AVE values to establish discriminant validity.

The highest squared bivariate correlation across all factors was 0.3969 (0.603^2). The smallest AVE across all factors was 0.410 [*(22) Iterative & Incremental Development*]. Therefore, discriminant validity is supported for all factors. However, given the high rate of false positives associated with this established validation method (Henseler et al., 2014; Rönkkö & Cho, 2020), we additionally applied the procedure recently developed by Rönkkö and Cho (2020). Specifically, we compared the upper limit of the 95% confidence interval (*CI*) of each inter-construct covariance (see Table S3 in the supplementary materials) with the recommended tolerance value of 0.81[1].

The discriminant validity of all factors in Model 3 was confirmed once again. However, the upper limits of the 95% *CI*s of the inter-construct covariance for the factor combinations [*(3) Mission-Driven Development* and *(9) Shared Product Understanding*], [*(7) Team-Based Planning* and *(14) Decentralized Decision-Making*], and [*(7) Team-Based Planning* and *(28) Review Practice*] are above the threshold of 0.8 (but below 0.83). According to Rönkkö and Cho (2020), an upper confidence limit below 0.8 provides no evidence of issues with discriminant validity, whereas a value up to 0.9 indicates only a marginal concern that does not invalidate the assumption of discriminant validity.

4.2.3 Discussion Study 1

This first study aimed to assess the validity of the QASE by empirically evaluating the theory-derived 30-factor measurement model. The CFA results indicated a very good fit between the hypothesized model and the observed data across all fit indices, providing strong support for the proposed 30-factor structure.

Model 3 was identified as the optimal solution in terms of reliability, validity, and parsimony. Of the 30 factors, 22 exhibited very good psychometric properties (Nájera Catalán, 2019; Nunnally, 1978), with reliability coefficients above 0.8 and AVE above 0.6 (see Table 2). Five factors exhibited good psychometric properties (Nájera Catalán, 2019; Nunnally, 1978), with reliability coefficients above 0.7 and AVE above 0.5. The remaining three factors *(13) Role Homogeneity, (22) Iterative & Incremental Development,* and *(25) Timeboxing* exhibited weak psychometric properties with an AVE between 0.4 and 0.5.

An AVE of 0.5 is generally regarded as evidence of adequate convergent validity. However, using this threshold as the sole criterion for rejecting convergent validity remains controversial (Cheung et al., 2024; Rubia, 2019). Some authors therefore recommend using additional criteria to assess convergent validity when the AVE falls below the threshold of 0.5 (Cheung et al., 2024; Cheung & Wang, 2017; Fornell & Larcker, 1981; Rubia, 2019). For example, Fornell and Larcker (1981) propose that, in such instances, CR as a less stringent criterion of reliability should be considered. They argue that convergent validity is acceptable if the CR exceeds the threshold of 0.6. Similarly, Cheung and Wang (2017) conclude in their simulation study that a construct demonstrates convergent validity if the AVE is not substantially below 0.5 and all factor loadings of the construct are also above 0.5. Consequently, considering these additional criteria, the convergent validity of the three factors *(13) Role Homogeneity, (22) Iterative & Incremental Development,* and *(25) Timeboxing* can be considered acceptable despite their AVE values falling below the recommended threshold of 0.5.

Taken together, the results of this first study indicate satisfactory construct validity, including convergent and discriminant validity, of all 30 factors, despite three factors falling slightly below the recommended AVE threshold. However, there is potential for improvement in the measurement accuracy of three factors: *(13) Role Homogeneity, (22) Iterative & Incremental Development,* and *(25) Timeboxing*. With AVE values of 0. 495, 0.410, and 0.448, respectively, these factors explain slightly less than half of the variance in their indicators. Further research is needed to improve the operationalization of these three factors through the development and validation of additional measurement items.

5 Study 2: Measurement Model Refinement

5.1 Methods

To address the weaknesses of the QASE identified in the first study, we developed new items for the three factors

[1] As recommended by Rönkkö & Cho (2020), the Reference-Group Method (Schweizer et al., 2019) was used as the scaling method for the estimation of the factor covariance

with AVE values below 0.5: *(13) Role Homogeneity, (22) Iterative & Incremental Development*, and *(25) Timeboxing*. Although a reliability coefficient of 0.7 is generally regarded in the literature as a good indicator of a construct's internal consistency (Hair et al., 2011; Nájera Catalán, 2019; Peterson & Kim, 2013), Nunnally (1978) (also see Lance et al. (2006)) recommends a value of 0.8 as desirable for further scale development and indicative of a mature measurement instrument. We, therefore, also developed new items for all factors with a reliability coefficient below 0.8 to enhance the psychometric properties of the QASE. These include the factors *(1) Pair Programming/Modeling, (4) Short Release Cycles, (8) Feature-Driven Development, (11) Collective Ownership*[2]*, (27) Open Workspace*, and *(29) Test-Driven Development*. Using newly developed items, unused items from our initial pool of 221, and selected items from the first study (those with the highest factor loadings), we generated a new pool of 55 measurement items designed to enhance the psychometric properties of these nine agility dimensions. The assignment of the items to the respective nine agility dimensions is shown in Appendix A (Model M4).

We conducted a second data collection through an online survey from October 29 to December 31, 2022, using Unipark. As with the initial survey, participants were informed beforehand of the anonymity and confidentiality of the data collection, that there were no right or wrong answers, and that the questions did not relate to their supervisor or the company where they worked. We again designed the survey to provide participants with the evaluated degree of agility in their project at the end and informed them of this at the beginning of the survey. The second study was pre-registered on OSF (https://osf.io/ays5f).

A total of 445 individuals accepted the invitation to participate in the survey. 289 began the questionnaire, 128 discontinued participation, and 161 completed the survey. After data cleaning, eight records were excluded. The remaining dataset contained 26.8% missing values. These were imputed using the same procedure as in the first study. Thus, all missing values were imputed except for one dataset, resulting in a final sample size of $N = 152$ for the data analysis.

The 152 participants were distributed across 36 different countries. India (28.9%) represented the largest group, followed by the USA (11.8%) and Germany (5.9%). Scrum was the most widely used agile method (55.3 %), followed by Agile Modeling (10.5%), Feature-Driven Development (5.9%), and Test-Driven Development (5.9%). 10.5% of the participants reported using a combination of the listed ten agile methods. 5.3% of the participants worked entirely from home, 7.9% entirely in the office, and 65.7% alternated between working from home and in the office. 21.1% did not provide any information. The largest groups of participants in terms of project roles were Programmers (50%), Software Architects (7.9%), Scrum Masters (5.3%), Team Leaders (5.3%), and Testers (3.9%).

As in the first study, we conducted a CFA using the robust diagonally weighted least squares (DWLS) estimator. This estimation method is recommended when the data do not follow a multivariate normal distribution and the sample size is relatively small (Li, 2016; Mîndrilă, 2010; Rhemtulla et al., 2012).

Following the same procedure, we tested three models that differed only in the number of items per factor: the hypothesized model (**Model 4**), an alternative model (**Model 5**) in which only the items with the highest internal consistency reliability (McDonald's ω) for each factor were retained, and **Model 6,** which retained only three items per factor. In Model 6, the retained items were selected to maximize the internal consistency reliability of each factor. The assignment of the items to the respective models is presented in Appendix A (see Models M4, M5, and M6).

5.2 Results

All three models demonstrate a good model fit (Hu & Bentler, 1999) across all fit indices, as shown in Table 3. Nevertheless, Model 6 exhibits the best overall model fit. The factor loadings ranged in Model 4 from 0.322 to 0.873, in Model 5 from 0.422 to 0.883, and in Model 6 from 0.497 to 0.936 (see Table S4 in the supplementary materials). Thus, all factor loadings in Model 6 exceeded the recommended threshold of 0.4 (Hulland, 1999).

Table 3: Goodness-of-fit indices of the tested models in Study 2

Model	χ^2	*df*	χ^2/df	*p*	CF	RMSEA	SRMR
Model 4	1888.170	1394	1.354	< .001	.977	.059	.077
Model 5	1300.085	909	1.430	< .001	.984	.054	.073
Model 6	450.083	288	1.563	< .001	.995	.036	.061

Note: For all models, $N = 152$. CFI – Comparative Fit Index; RMSEA – Root-Mean-Square Error of Approximation; SRMR – Standardized Root-Mean-Square Residual. **Model 4:** hypothesized model (55 items), **Model 5:** model with the highest factor-level McDonald's ω (45 items), **Model 6:** three-item-per-factor model (27 items).

Following scale purification, Model 5 exhibited the highest reliability among the three models (see Table 4). However, the largest difference in reliability between Models 5 and 6 was only 0.052 (see *(29) Test-Driven Development*), which is considered acceptable given the higher parsimony of Model 6. Five of the nine factors in Model 6 exhibited reliability coefficients above the targeted value of 0.8. The remaining four factors *(1) Pair Programming/Modeling, (8)*

[2] The inclusion of the factor *(11) Collective ownership* among the factors with a reliability coefficient of less than 0.8, although it had previously exceeded the threshold of 0.8 in the first study, is due to a change in the item adjustment method that was made after the data for Study 2 were collected. This adjustment consisted of prioritizing the items with the highest reliability coefficients rather than those with the highest factor loadings. As a result, the reliability of most factors, including *(11) Collective Ownership*, has improved and is now above 0.8.

Feature-Driven Development, (22) Iterative & Incremental Development, and *(29) Test-Driven Development*, however, exceeded the commonly accepted threshold of 0.7 (Hair et al., 2011; Nájera Catalán, 2019; Peterson & Kim, 2013). Compared to Models 4 and 5, Model 6 exhibited the highest AVE across all factors, except for factors *(25) Timeboxing* and *(27) Open Workspace.* All factors in Model 6exceeded the targeted AVE of 0.5, with the exception of *(8) Feature-Driven Development*.

Table 4: Congeneric and construct reliability and convergent validity of the tested factors in Study 2

		McDonald's ω			CR			AVE			
N°	Factor	Model 4	Model 5	Model 6	Model 4	Model 5	Model 6	Model 4	Model 5	Model 6	Model 3
1	Pair Programming/Modeling	.783	.783	.752	.809	.809	.784	.464	.465	.555	.513
4	Short Release Cycles	.825	.825	.809	.867	.866	.837	.521	.521	.631	.567
8	Feature-Driven Development	.710	.710	.702	.735	.732	.703	.366	.366	.464	.535
11	Collective Ownership	.825	.825	.783	.872	.872	.826	.533	.533	.613	.668
13	Role Homogeneity	.824	.824	.809	.851	.851	.825	.534	.533	.613	.495
22	Iterative & Incremental Development	.748	.778	.734	.809	.812	.761	.357	.468	.518	.410
25	Timeboxing	.742	.820	.820	.820	.862	.855	.386	.675	.663	.448
27	Open Workspace	.832	.845	.837	.891	.898	.866	.580	.688	.683	.507
29	Test-Driven Development	.736	.736	.712	.803	.803	.751	.408	.408	.502	.569

Note: For all factors, $N = 152$. CR _ Congeneric reliability; AVE _ Average Variance Extracted. **Model 4**: hypothesized model (55 items), **Model 5**: model with the highest factor-level McDonald's ω (45 items), **Model 6**: three-item-per-factor model (27 items), **Model 3** (Study 1): the corresponding nine factors from Study 1.

5.3 Discussion Study 2

The second study aimed to improve the psychometric properties of the QASE by refining the operationalization of factors that exhibited weaker reliability and validity in the first study. We therefore developed new items for these factors and re-examined their reliability and validity.

Of the nine re-operationalized factors, six could be improved compared to Model 3 (Table 4). Factors *(11) Collective Ownership* and *(29) Test-Driven Development* remained above the threshold of 0.7 for reliability. However, there was no improvement compared to the initial study. In contrast to the first study, factor *(8) Feature-Driven Development* falls below the recommended AVE threshold of 0.5.

These results suggest that the psychometric properties of QASE can be improved by replacing the first developed items of the factors *(1) Pair Programming/Modeling, (4) Short Release Cycles, (13) Role Homogeneity, (22) Iterative & Incremental Development, (25) Timeboxing, and (27) Open Workspace* in Model 3 with the newly developed items from Model 6. The proposed combination (Model 7) is presented in Appendix A (Model M7). However, further studies should aim to re-evaluate the factor structure and discriminant validity of Model 7, as the reliability and validity of the combined factors were validated in two different models. Although the second study primarily aimed to improve the operationalization of the nine factors rather than replicate the first study, the factor analysis results supported the factor structure identified in the first study for these factors.

6 General Discussion

6.1 General Discussion of the Results

Given the lack of consensus on what constitutes agility and the resulting diverse conceptualizations of the construct in the literature, this work responds to the growing call for a psychometric validation study aimed at developing a valid measurement instrument that can be used consistently across different studies (Abrahamsson et al., 2009; Ågerfalk et al., 2009; Chronis & Gren, 2016; Junker et al., 2021; Nguyen et al., 2024; Rietze & Zacher, 2022; Tripp et al., 2018). Considering the divergent evolution of the concept over the past decade, we, as suggested by various authors of the Manifesto for Agile Software Development, opted to revisit the roots that initially brought this concept to the mainstream, aiming to restore clarity (Fowler, 2006; Hohl et al., 2018; Thomas, 2014). Historically, the concept of Agile Software Development emerged from a set of pioneer software development methods, and the agile values and principles that form the cornerstone of this concept today emerged from an initiative by software professionals to articulate the commonalities of these methods as a theoretical core. Accordingly, similar to the agile values and principles, we conceptualized Agile Software Development as a common characteristic of these pioneer agile methods;

however, at the level of their techniques to overcome the interpretive latitude inherent in the agile values and principles. This resulted in a second-order formative construct with 30 first-order reflective dimensions. Through this conceptualization, the fundamental challenge in measuring this construct, in terms of the lack of consensus on a definition of agility and the high level of abstraction of the agile values and principles, could be overcome. As a result, a theoretically sound measurement model (QASE) was derived that addresses the limitations of previous research in this area, as discussed in the theoretical section of this paper. The CFA provided empirical support for the proposed 30-factor measurement model, indicating that the hypothesized factor structure is consistent with the observed data. The analysis of the psychometric properties of the measurement instrument, in terms of validity and reliability, provided strong evidence for the validity and reliability of most factors in the first study, with the exception of three factors: *(13) Role Homogeneity, (22) Iterative and Incremental Development,* and *(25) Timeboxing*. These were refined in a second study, which enhanced their reliability and validity, resulting in a final model (Model 7, Appendix A, M7) with good to very good psychometric properties.

At first glance, the QASE appears similar to instruments that measure agile practices (e.g., Junker et al., 2022; So & Scholl, 2009; Tripp et al., 2016). However, the QASE does not refer to individual practices of one or more methods or the agile practices of a single method. Instead, similar to the agile values and principles, it reflects shared characteristics across agile methods based on their agile techniques, including processes, roles, and practices. Therefore, in contrast to other instruments, we do not refer to the QASE as an instrument for measuring agile practices, but as a questionnaire to assess agility in software engineering. However, the extent to which agility can be reduced to a list of its techniques remains controversial in the literature. This debate is often framed in terms of "being agile versus doing agile." In this context, some authors (e.g., Duka, 2012; Eilers et al., 2020; Horlach & Drechsler, 2020; Rahman et al., 2018) draw attention to what Baham and Hirschheim (2021) call the "pitfalls of reductionism". This perspective suggests that merely following agile practices (doing agile) differs from "being agile". The latter "comprises everything that affects the employees in their work process by using agile methods" (Eilers et al., 2020, p. 2) and is more than an implementation of agile techniques à la carte (Baham & Hirschheim, 2021). In view of this debate, the question naturally arises as to what extent the QASE measures adherence to agile practices rather than agility per se. Numerous studies have provided empirical evidence for a relationship between agile practices and various outcomes such as job satisfaction (e.g., Kanwal & Khurram, 2022; Tripp et al., 2016), well-being (e.g., Rietze & Zacher, 2022; Syed-Abdullah et al., 2006), or project success (e.g., Sandstø & Reme-Ness, 2021; Serrador & Pinto, 2015). It is therefore well established that agile techniques have the potential to contribute to positive outcomes. The essence of this discourse, therefore, primarily concerns the question of which agile practices or combinations of practices should be employed and how they should be implemented to achieve agility (Baham & Hirschheim, 2021). Indeed, the incorrect or inappropriate implementation of agile practices can lead to counterproductive results, as discussed by Cagle (2019). Similarly, omitting "key" agile techniques or inappropriate method tailoring can lead to non-agility or partial agility (Baham & Hirschheim, 2021). For instance, an ineffective stand-up meeting may be perceived by the team as a mechanism of control rather than as an opportunity for information exchange. Likewise, a two-week sprint may be inappropriate where a four-week sprint better fits the project context, or a requirement for continuous testing without sufficient automation support may lead to rejection due to the effort involved.

In this regard, the QASE addresses this criticism through three design features. First, the QASE is not a random assortment of agile practices; rather, it reflects the commonalities of the pioneer agile methods and thus embodies the theoretical core of agile software development (Baham & Hirschheim, 2021; Hohl et al., 2018; Strode, 2005). Second, as discussed in the theoretical section of this paper, although agile values and principles form the common foundation of agile methods, differences still exist among the various existing agile methods due to their context-dependent nature. Baham and Hirschheim (2021), therefore, argue that when attempting to measure agile software development, it is essential to consider the project context, as the agile phenomenon is a highly diverse, adaptable, contextual, and situated practice. By clustering agile techniques from various methods, we address this challenge and establish a context-independent conceptualization of the related techniques. For instance, the cluster "*Competence Coverage*" is derived from the requirement of a senior developer in DSDM, a chief programmer in FDD, a coach in XP, or a senior design programmer in CFM. By aggregating these context-dependent techniques into "*Competence Coverage*," the related items of the QASE do not assess whether a senior design programmer or coach is present in the team, but whether the team has the necessary (from the team perspective) support to perform its work effectively. (See items related to *Competence Coverage,* Appendix A). Third, we operationalized each agile technique as a reflective construct. This approach ensures that the focus is not on the adherence to the techniques themselves, but rather on the extent to which the techniques were able to unfold their effect on the team members. For instance, the factor *Short Release Cycles* is not measured by assessing the duration of the release or whether it lasts 2 or 3 weeks, as recommended in some methods. Instead, it is measured by assessing how helpful the duration of the release (regardless of its length) was for the team, for example, to process (customer) feedback with little effort (See items related to *Short Release Cycles,* Appendix A). In this way,

the QASE captures the agility reflected in the team's actual way of working, regardless of the nomenclature of techniques used, whether an agile technique is implicitly or explicitly applied, or whether the project actively strives for agility by applying an agile method.

The QASE, therefore, reflects the defining characteristic of agile values and principles, namely their role as the common characteristics of all agile methods, while remaining context-neutral, making it applicable to agile approaches beyond the considered pioneer agile methods. For instance, a systematic literature review of agile software development projects (Rath et al., 2025) reveals that the pioneer agile methods and their techniques have been adapted over the past decade to respond to changes in software development, such as the ever-increasing size of agile projects. This has led to the scaling of agile software development, thereby broadening the original scope of the concept, which now extends to accommodate globally distributed teams rather than relying solely on face-to-face communication. In this context, new roles such as "Code Buddy", "Technical Area Responsible", or practices such as "Sync Meeting" were introduced (Vallon et al., 2018). However, at the heart of these new techniques lies the theoretical core of Agile Software Development, from which the QASE was built. For example, the role of "Code Buddy" and "Technical Area Responsible" coincides with the agile dimension of "*Competence Coverage,*" and the practice of "Sync Meeting" with the agile dimension of "*Regular Meeting*". Through the reflexive operationalization of the respective agile dimensions, the QASE accommodates such contextual adaptations. For instance, the agile dimension of "*Open Workspace*" is not captured by assessing whether face-to-face communication takes place, but rather whether communication is facilitated effectively, regardless of the specific medium. Thus, the application of the QASE is not limited to projects explicitly using one of the pioneer agile methods or any method labeled as agile but can also be applied in the context of hybrid methods or any way of working that implicitly embodies the agile philosophy.

From a broader perspective, the developed instrument captures core structural characteristics of agility in software development. Several of these dimensions—such as Iterative Planning, Customer Involvement, and Empowerment—are also reflected in related theoretical perspectives, including Lean Thinking, which similarly emphasizes aspects such as iterative feedback cycles, customer-centric value creation, and decentralized decision-making (Layik, 2021; Poppendieck & Poppendieck, 2003). This conceptual alignment strengthens the theoretical robustness of the instrument, as it indicates that the measured dimensions are not specific to a single agile method or framework, but instead capture more general principles of modern software development. As such, the instrument supports a more generalizable operationalization of agility, enabling its application across different methodological contexts and providing a more stable foundation for empirical research on software development practices.

Revisiting the roots of Agile Software Development and drawing from the original pioneer methods may raise the concern that our conceptualization of Agile Software Development overlooks 20 years of research in this field, and that a systematic literature review of Agile Software Development would provide a better approach to understanding and capturing this concept. Indeed, prior research has undoubtedly contributed valuable insights into our understanding of the Agile Software Development phenomenon, and the differences between these studies can be interpreted as reflecting different perspectives of the concept. However, previous literature reviews on the definition of Agile Software Development indicate that its varied interpretations stem not only from the different theoretical perspectives of the concept, but are also associated with a stretching of the concept over time (Laanti et al., 2013; Schirrmacher & Schoop, 2018). The phenomenon of concept stretching is controversial among scientists. While some scholars advocate for concept stretching as a means of enriching our understanding of a concept (e.g., Welch et al., 2016), others argue that it causes dilution and confusion and therefore reject it (e.g., Osigweh, 1989; Sartori, 1970). In this context, as described by Solinger et al. (2024), three design traditions for conceptualization have been established in management literature. The constructionist definition of concepts, based on the fundamental idea that "a concept is what people say it is"; the empiricist definition of concepts, according to which "a concept is what it 'does'"; and the rationalist definition of a concept, defined as "concept as constitutive of theory" (Solinger et al., 2024, pp. 4–5). While these different positions are often viewed as antagonistic in the literature, Solinger et al. (2024) advocate drawing on these different positions as an instrument to maintain the health and vitality of the literature. They introduce the concept of conceptual maintenance, which they define as "targeted corrections of meaning deemed necessary by (…) authors and reviewers to secure theoretical novelty and/or usefulness of knowledge around an (…) subject" (Solinger et al., 2024, p. 11). Conceptual maintenance, thereby, consists of a deliberate shift in the conceptual inquiry from universality to particularity, from metaphysical to experiential investigation or vice versa, depending on the state of the literature (Solinger et al., 2024). However, they emphasize that "crucial to the idea of maintenance is that creativity on the part of a scholar is combined with reflexivity (Weick, 1999) on what the literature in question needs in order to move forward" (Solinger et al., 2024, p. 11). Given that the concept of Agile Software Development is becoming increasingly equivocal, leading to growing confusion in the literature, we consider it an important and necessary scientific contribution to return to its conceptual foundations to restore clarity, thus re-establishing a solid foundation for a better understanding of the concept. Furthermore, despite multiple suggestions to revise the

Manifesto for Agile Software Development (e.g., Barber, 2020; Denning, 2015; Jeffries, 2025; Schlein, 2022), none have gained traction. The Manifesto for Agile Software Development (Beck et al., 2001b) continues to be the fundamental consensus recognized in much of the literature on agility in information systems (Schirrmacher & Schoop, 2018), and therefore represents one of the most established foundations for understanding the construct.

6.2 Theoretical implications

The development and validation of our measurement instrument complement existing literature in several ways. First, it prompts reflection on the increasing ambiguity of the concept of agility in the literature, resulting from the proliferation of new interpretations, and therefore calls for greater consensus in research on agility through improved conceptual coherence across studies in this field. The QASE itself contributes to this call in two respects. It provides a thorough measurement model of Agile Software Development applicable across various contexts, thereby facilitating more cohesive and cumulative research in the field. Moreover, it consolidates research on the measurement of agility. Its theoretically grounded conceptualization addresses the limitations of previous work, while its operationalization integrates items from multiple existing measurement instruments. Consequently, the QASE represents more than just another tool in the existing body of agile measurement instruments and responds to the call by Gren et al. (2015) to validate and consolidate existing instruments rather than develop new ones.

Second, the analysis of individual agile techniques (e.g., Ghimire & Charters, 2022; Rietze & Zacher, 2022; Tripp et al., 2016; Uraon et al., 2023) or comparison of agile projects with non-agile projects (e.g., Gemino et al., 2020; Kakar, 2017; Kassab et al., 2018; Lalić et al., 2022) represent the two common research designs used to investigate the concept of agility. While the former does not provide a holistic understanding of the concept, the latter is associated with potential misclassification bias. The results of our research address both limitations and provide a measurement instrument that enables new empirical research, such as investigating the potential interactions among agile techniques to promote agility, and supports more robust research designs that conceptualize agility as a continuous rather than dichotomous construct, thereby reducing the risk of misclassification bias.

Fourth, research on agility has suffered from a lack of a valid measurement instrument that enables the comparability of research results (Abrahamsson et al., 2009; Tripp et al., 2018). In this respect, the QASE, through its conceptualization and operationalization, constitutes a method-independent questionnaire that is suitable for use across agile methodologies, thereby enabling broad application in research and facilitating the comparability of results, regardless of working methods, team composition, project type, or context.

Fifth, the results of our study contribute a significant step toward a more nuanced understanding of the concept of agility in software engineering and its applicability to other domains. As agile methods are customized in practice to suit the unique needs of each project, Baham and Hirschheim (2021) highlight the need for a better understanding of how different agile practices interact and the consequences of omitting techniques by method tailoring. By providing a comprehensive list of agile techniques, the QASE offers a foundation for investigating, for example, which techniques contribute to specific outcomes, how agile techniques interact with one another, or which combinations of techniques are most effective. Beyond facilitating empirical research on individual practices and their relationships, the QASE also contributes to the broader theoretical development of Agile Software Development. Tripp et al. (2018) emphasize that identifying a core theory of Agile Software Development represents a fundamental prerequisite for advancing agile research and establishing a generalized understanding of agility. As one of the most mature and established domains of agility research (Hoda et al., 2018), Agile Software Development provides an appropriate starting point for investigating the mechanisms underlying agile outcomes and deriving insights that may eventually inform other domains. Although the concept of agility in practice has long moved beyond the manufacturing industry and software development to Enterprise Agility, which extends agile principles beyond software teams to encompass entire organizations, focusing on structure, leadership, and culture (Atanassova et al., 2025; Bennett & Lemoine, 2014; Magistretti & Trabucchi, 2025; Pinho et al., 2022), a corresponding generalized theoretical framework remains a significant desideratum in current research (Nguyen et al., 2024; Walter, 2021; Yang et al., 2025). By uncovering the core elements of Agile Software Development and making them measurable, the QASE provides a foundation for investigating how and why Agile Software Development contributes to positive outcomes, and for deriving insights that may inform the broader understanding of agility beyond the software development (Baham & Hirschheim, 2021; Drechsler & Ahlemann, 2015).

6.3 Practical implications

From a practical perspective, our study provides an overview of the common techniques that characterize agile working in software engineering, thus introducing a method-independent set of dimensions that captures the theoretical core of Agile Software Development. Organizations can therefore use the QASE dimensions as a reference when designing or tailoring their way of working, enabling them to focus on agile elements grounded in the common foundation of established agile methods. Furthermore, the QASE represents a theoretically sound

and empirically valid instrument for evaluating the alignment of working methods with the agile working philosophy and identifying problem areas where improvements may be required.

In industrial practice, assessments of agility are often based on informal or subjective perceptions. The QASE, however, enables these perceptions to be operationalized in a more systematic, transparent, and comparable manner. In particular, it facilitates meaningful comparisons across teams, projects, and time periods. These comparative capabilities are especially important for identifying effective practices, monitoring progress over time, and evaluating the impact of process improvement initiatives. The QASE can be integrated into iterative feedback mechanisms such as retrospectives or inspect-and-adapt workshops. Within these contexts, it can be used to systematically elicit structured feedback on aspects of the development process that require refinement. For example, aggregated QASE scores can be used as a Key Performance Indicator to characterize a team's overall agility profile and identify specific dimensions where performance is comparatively weaker. By highlighting such dimensions, the instrument enables teams to prioritize improvement efforts, track changes over time, and evaluate the effectiveness of implemented interventions.

While the QASE primarily measures agility at the team and project levels — as these are the primary units where agile techniques are typically enacted and observable — the instrument can also be applied at the organizational level as an aggregated measure to evaluate how deeply agile principles and techniques are embedded across multiple teams within an organization. Accordingly, the instrument supports diagnostic analysis at multiple levels of granularity (project, team, and organization), thereby contributing to evidence-informed decision-making and continuous improvement processes.

Finally, while the concept of the reflective measurement model used to operationalize the respective agile techniques is long-established in research, its application in practice can also contribute to improving the exercise of agile software development. For instance, rather than strictly adhering to the textbook application of the technique or agile method, practitioners should reflect on the outcomes that these techniques are intended to achieve and whether those outcomes have been realized through their actual work approach.

6.4 Limitations and Future Works

Although the results of the statistical analyses supported the validity of our theoretically derived model and thus our conceptualization, there remains scope for further analysis of the overall measurement model, continued development and refinement of the subscales, and future replications. For instance, in contrast to 30 first-order reflective models, no statement could be made about the error term of the second-order formative construct. Thus, the question remains as to what percentage of the variance in agility is explained by the 30 factors. Agility was conceptualized as a second-order formative model composed of 30 dimensions. The error term for such a model cannot be estimated unless an alternative measure of the construct is included (Sarstedt et al., 2022). Since there is no generally valid (alternative) measurement instrument that can be used for this purpose, the measurement error of the entire measurement model could not be estimated. However, Diamantopoulos (2006) points out that the error term in formative models, unlike reflective models, is not caused by the indicators in the model, but by the non-modeled causes of the construct. For this reason, a formative model is free of error terms once all aspects of the construct have been considered in the model specification (Diamantopoulos, 2006). Regarding the development of the QASE, the broadest possible set of commonalities across the ten analyzed agile methods was considered (i.e., each technique is prescribed in at least two methods), so it can be expected that little unexplained variance should remain at the second-order construct level. An additional analysis of the measurement model for multicollinearity to assess potential factor overlap (Diamantopoulos, 2006; Sarstedt et al., 2022) was not conducted, because potential overlap among the factors had already been examined through the analysis of factor covariances during the CFA.

The sample size (N = 418) used to evaluate the QASE, together with the relatively high percentage of cases with imputed values (35.7%), represents a further limitation of our study and raises questions about the replicability of the results. Missing values were addressed using person-mean imputation, a commonly applied approach for handling item-level missing values in multi-item scales with a low level of missingness, as it preserves the sample size while reducing the bias typically associated with listwise deletion (Bono et al., 2007; Enders, 2022; Parent, 2013). Furthermore, it has been shown to produce approximately unbiased parameter estimates while maintaining the underlying factor structure and exerting only a minimal impact on overall validity (Huisman, 2000; Peyre et al., 2011; Roth et al., 1999; Schafer & Graham, 2002). Consistent with these findings, Chen et al. (2012) demonstrated in a simulation study that, even with missing value rates exceeding 30% and under conditions comparable to those of the present study, the application of an appropriate imputation procedure (in this case, mean imputation) has only a marginal effect on CFA validity. Consequently, the characteristics of our data are unlikely to substantially compromise the replicability and validity of our results. However, they limited the scope of the analysis of the psychometric properties of our measurement instrument, which leaves room for further investigation and replication. For instance, the collected data did not follow a multivariate normal distribution and were therefore unsuitable for model estimation using conventional maximum likelihood (ML). Therefore, the model was estimated using DWLS (Li, 2016; Mîndrilă, 2010). Further

research could therefore seek to replicate the findings using multivariate normally distributed data, thereby permitting the application of Maximum Likelihood Estimation (MLE) as a conventional approach to structural equation modeling.

A further limitation concerns the assessment of the measurement invariance of the QASE. This could not be evaluated due to the small sample size. Consequently, further work should investigate the equivalence of the model in different countries, project roles, and project types (e.g., pre-development project, customer project, front-end/back-end web development, embedded software development, etc.). Additionally, the QASE was designed as a method-independent instrument for assessing agility in software development, regardless of the organizational context or the specific agile approach employed. To ensure this general applicability, we followed the conceptual rationale underlying the development of the Agile Manifesto by considering the same ten pioneering agile approaches referenced by its authors to identify the common theoretical core of agile software development. While the empirical validation already included a highly heterogeneous sample representing a broad range of software development contexts and agile approaches, future research should further examine the applicability and validity of QASE in software development settings employing agile approaches beyond the ten pioneering approaches considered in this study, including more recent approaches such as large-scale agile approaches, as well as in additional organizational contexts.

Beyond measurement invariance, the objectivity of the QASE should also be further investigated. The QASE assesses project agility based on the self-reported perceptions of project members, which may be influenced by their roles in the project. Consequently, further research should assess the objectivity of the QASE. This can be achieved by examining test–retest reliability and evaluating the QASE based on data from multiple participants within the same project. However, when assessing the objectivity of the QASE, it is essential to distinguish between projects without formal role differentiation and projects with formally defined roles, in which perceptions may differ across roles within the same project. This distinction is particularly relevant since one dimension of agility captured by the QASE refers to the absence of rigid role differentiation and emphasizes shared responsibility within the team. Accordingly, analyses of measurement invariance and objectivity are expected to reveal greater variation in assessments across respondents in projects with distinct role structures, whereas projects without formal role differentiation should demonstrate greater agreement among respondents.

Finally, the evaluation of the refined QASE in the second study was limited to the modified factors and did not include the other dimensions. Consequently, further work should re-evaluate the resulting Model 7 (Appendix A, M7) to confirm the validity of all dimensions when combined.

Despite its limitations, this research makes a substantial contribution to the ongoing discourse on what constitutes agility and how it can be measured. While a pragmatic consensus in this debate may be to accept, as Kuhrmann et al. (2021, p. 14) put it, that ““agility” is in the eye of the beholder and therefore a subjective concept”, our findings suggest that this subjectivity does not preclude the existence of a shared theoretical foundation. Rather, the pioneers of Agile Software Development established a common basis on which theories can be built, and much remains to be explored and learned. As the authors of the Manifesto for Agile Software Development emphasize: “What is new about agile methods is not the practices they use, but their recognition of people as the primary drivers of project success” (Highsmith & Cockburn, 2001, p. 122). With the QASE, we provide a theoretically sound and empirically valid measurement instrument to capture the concept, thereby facilitating the formulation and testing of hypotheses, the application of theories, and the generation of more reliable conclusions about the processes underlying the positive outcomes associated with this phenomenon. The challenge of measuring agility may partly explain the relatively limited empirical research on the concept, considering its popularity (Rytkönen et al., 2025). It is, therefore, our hope that our study and its results will stimulate empirical research in the context of agile development.

Reflecting on the need for more empirical research and greater theoretical and methodological rigor as the major bottlenecks in agile research at the time, Dingsøyr et al. (2012) conclude their contribution to the tenth anniversary of the Manifesto for Agile Software Development by emphasizing that “it is important to remember that the field can only mature and evolve as a scientific discipline if efforts are made to establish a robust theoretical framework for conducting research on agile development.” Since then, considerable progress has been made, but there is still more to be done toward achieving this goal (Baham & Hirschheim, 2021; Tripp et al., 2018). We hope that our work represents a step in this direction.

References

Abrahamsson, P., Conboy, K., & Wang, X. (2009). ‘Lots done, more to do’: The current state of agile systems development research. *European Journal of Information Systems*, *18*(4), 281–284. https://doi.org/10.1057/ejis.2009.27

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods: Review and analysis (VTT Publications No. 478). *VTT Technical Research Centre of Finland*. https://www.vttresearch.com/sites/default/files/pdf/publications/2002/P478.pdf

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New directions on agile methods: A comparative analysis. *Proceedings of the 25th*

International Conference on Software Engineering, 244–254. https://doi.org/10.1109/icse.2003.1201204

Acuña, S. T., Lopez, M., Juristo, N., & Moreno, A. (1999). A process model applicable to software engineering and knowledge engineering. *International Journal of Software Engineering and Knowledge Engineering*, *9*(5), 663–687. https://doi.org/10.1142/s0218194099000358

Ågerfalk, P. J., Fitzgerald, B., & Slaughter, S. A. (2009). Introduction to the special issue—Flexible and distributed information systems development: State of the art and research challenges. *Information Systems Research*, *20*(3), 317–328. https://doi.org/10.1287/isre.1090.0244

Agile Alliance. (2015, June 29). *Agile essentials: Agile101 What is Agile?* https://agilealliance.org/agile101/

Akkaya, B., Panait, M., Apostu, S. A., & Kaya, Y. (2022). Agile leadership and perceived career success: The mediating role of job embeddedness. *International Journal of Environmental Research and Public Health*, *19*(8), 4834. https://doi.org/10.3390/ijerph19084834

Ambler, S. (2002). *Agile modeling: Effective practices for extreme programming and the unified process* (1st ed.). Wiley.

Amrit, C., & Meijberg, Y. (2018). Effectiveness of test-driven development and continuous integration: A case study. *IT Professional*, *20*(1), 27–35. https://doi.org/10.1109/MITP.2018.014121554

Ariza, H. M., Silva, L. F. W., & Contreras, L. A. L. (2020). Descriptive analysis of the agile methodology extreme programming (XP) for its implementation in software development. *International Journal of Engineering Research and Technology*, *14*(10), 999–1004.

Atanassova, I., Bednar, P., Khan, H., & Khan, Z. (2025). Managing the VUCA environment: The dynamic role of organizational learning and strategic agility in B2B versus B2C firms. *Industrial Marketing Management*, *125*, 12–28. https://doi.org/10.1016/j.indmarman.2024.12.008

Atzberger, A., Wallisch, A., Nicklas, S., & Paetzold, K. (2020). Antagonizing ambiguity – Towards a taxonomy for agile development. *Procedia CIRP*, *91*, 464–471. https://doi.org/10.1016/j.procir.2020.02.200

Azad, N., & Hyrynsalmi, S. (2023). DevOps critical success factors—A systematic literature review. *Information and Software Technology*, *157*, 107150. https://doi.org/10.1016/j.infsof.2023.107150

Baham, C., & Hirschheim, R. (2021). Issues, challenges, and a proposed theoretical core of agile software development research. *Information Systems Journal*, *32*(4), 103–129. https://doi.org/10.1111/isj.12336

Barber, M. (2020, May 14). An updated agile manifesto. *Medium*. https://medium.com/@markbarbs/an-updated-agile-manifesto-6aa5cf168101

Beck, K. (1999). *Extreme programming explained: Embrace change* (1st ed.). Addison-Wesley Longman.

Beck, K. (2003). *Test-driven development: By example*. Addison-Wesley Professional.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001a). *History: The Agile Manifesto*. https://agilemanifesto.org/history.html

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001b). *Manifesto for agile software development*. https://agilemanifesto.org/

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001c). *Principles behind the Agile Manifesto*. https://agilemanifesto.org/principles.html

Bennett, N., & Lemoine, G. J. (2014). What a difference a word makes: Understanding threats to performance in a VUCA world. *Business Horizons*, *57*(3), 311–317. https://doi.org/10.1016/j.bushor.2014.01.001

Bergmann, T. (2018). *The relationship between agile project management and project success outcomes* [Doctoral dissertation, University of Central Florida]. https://stars.library.ucf.edu/cgi/viewcontent.cgi?article=7210&context=etd

Bono, C., Ried, L. D., Kimberlin, C., & Vogel, B. (2007). Missing data on the Center for Epidemiologic Studies Depression Scale: A comparison of 4 imputation techniques. *Research in Social and Administrative Pharmacy*, *3*(1), 1–27. https://doi.org/10.1016/j.sapharm.2006.04.001

Bridgman, P. W. (1950). The operational aspect of meaning. *Synthese*, *8*(6–7), 251–259.

Brunet, F., Malas, K., & Fleury, D. (2021). A model of an agile organization designed to better manage the COVID-19 crisis. *Healthcare Management Forum*, *34*(2), 115–118. https://doi.org/10.1177/0840470420980478

Burgess, J. A. (1990). The sorites paradox and higher-order vagueness. *Synthese*, *85*(3), 414–474.

Cagle, K. (2019, August 23). *The end of agile*. Forbes. https://www.forbes.com/sites/cognitiveworld/2019/08/23/the-end-of-agile/

Chen, S.-F., Wang, S., & Chen, C.-Y. (2012). A simulation study using EFA and CFA programs based the impact of missing data on test dimensionality. *Expert Systems with Applications*, *39*(4), 4026–4031. https://doi.org/10.1016/j.eswa.2011.09.085

Cheung, G. W., Cooper-Thomas, H. D., Lau, R. S., & Wang, L. (2024). Reporting reliability, convergent and discriminant validity with structural equation modeling: A review and best-practice recommendations. *Asia Pacific Journal of Management*, *41*(2), 745–783. https://doi.org/10.1007/s10490-023-09871-y

Cheung, G. W., & Wang, C. (2017). Current approaches for assessing convergent and discriminant validity with SEM:

Issues and solutions. *Proceedings - Academy of Management*, *2017*(1), Article 12706. https://doi.org/10.5465/ambpp.2017.12706

Chronis, K., & Gren, L. (2016). Agility measurements mismatch: A validation study on three agile team assessments in software engineering. In H. Sharp & T. Hall (Eds.), *Agile Processes, in software engineering and extreme programming* (pp. 16–27). Springer International Publishing. https://doi.org/10.1007/978-3-319-33515-5_2

Churchill, G. A. (1979). A paradigm for developing better measures of marketing constructs. *Journal of Marketing Research*, *16*(1), 64–73. https://doi.org/10.2307/3150876

Cico, O., Jaccheri, L., Nguyen-Duc, A., & Zhang, H. (2021). Exploring the intersection between software industry and software engineering education—A systematic mapping of software engineering trends. *Journal of Systems and Software*, *172*, 110736. https://doi.org/10.1016/j.jss.2020.110736

Cockburn, A. (2000). *Writing effective use cases* (1st ed.). Addison-Wesley Professional.

Cockburn, A. (2002). *Agile software development*. Addison-Wesley Longman.

Cohard, P., & Messeghem, K. (2022). A measurement scale for agile orientation. *Systèmes d'information & Management*, *27*(1), 39–66. https://doi.org/10.3917/sim.221.0039

Cohen, D., Lindvall, M., & Costa, P. (2004). *An introduction to agile methods*. *Advances in Computers*, *62*, 1–66. https://doi.org/10.1016/s0065-2458(03)62001-2

Coltman, T., Devinney, T. M., Midgley, D. F., & Venaik, S. (2008). Formative versus reflective measurement models: Two applications of formative measurement. *Journal of Business Research*, *61*(12), 1250–1262. https://doi.org/10.1016/j.jbusres.2008.01.013

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, *20*(3), 329–354. https://doi.org/10.1287/isre.1090.0236

Conboy, K., & Fitzgerald, B. (2004). Toward a conceptual framework of agile methods. *Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research*, 37–44. https://doi.org/10.1145/1029997.1030005

Corrêa Rodrigues, A., & Mantovani Fontana, R. (2019). Evaluation of an agile maturity model: Empirical evidences for agility assessments. In G. S. Tonin, B. Estácio, A. Goldman, & E. Guerra (Eds.), *Agile methods* (pp. 49–62). Springer International Publishing. https://doi.org/10.1007/978-3-030-14310-7_4

Darwish, N. (2014). Enhancements in SCRUM framework using extreme programming practices. *International Journal of Intelligent Computing and Information Sciences*, *14*(2), 53–67. https://doi.org/10.21608/ijicis.2014.15773

De Barros, L. M., Tam, C., & Varajão, J. (2024). Agile software development projects – unveiling the human-related critical success factors. *Information and Software Technology*, *170*, 107432. https://doi.org/10.1016/j.infsof.2024.107432

De O Luna, A. J. H., & Marinho, M. (2023). Agile governance theory: A multi-scenario empirical assessment. *Observatorio de La Economía Latinoamericana*, *21*(6), 5767–5800. https://doi.org/10.55905/oelv21n6-135

Denning, S. (2015). Updating the Agile Manifesto. *Strategy & Leadership*, *43*(5). https://doi.org/10.1108/SL-07-2015-0058

Diamantopoulos, A. (2006). The error term in formative measurement models: Interpretation and modeling implications. *Journal of Modelling in Management*, *1*(1), 7–17. https://doi.org/10.1108/17465660610667775

Diamantopoulos, A., Riefler, P., & Zeugner‐Roth, K. P. (2008). Advancing formative measurement models. *Journal of Business Research*, *61*(12), 1203–1218. https://doi.org/10.1016/j.jbusres.2008.01.009

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, *85*(6), 1213–1221. https://doi.org/10.1016/j.jss.2012.02.033

Drechsler, A., & Ahlemann, F. (2015). Toward a general theory of agile project management—A research design. *ECIS 2015 Research-in-Progress Papers*. https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1024&context=ecis2015_rip

Duka, D. (2012). Agile experiences in software development. *Proceedings of the 35th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 692–697.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9–10), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006

Eilers, K., Peters, C., & Leimeister, J. M. (2022). Why the agile mindset matters. *Technological Forecasting and Social Change*, *179*, 1–14. https://doi.org/10.1016/j.techfore.2022.121650

Eilers, K., Simmert, B., & Peters, C. (2020). *Doing agile vs. being agile—Understanding their effects to improve agile work*. *ICIS 2020 Proceedings*, *6*. https://aisel.aisnet.org/icis2020/is_development/is_development/6

Enders, C. K. (2022). *Applied missing data analysis* (2nd ed.). The Guilford Press.

Fewell, J. (2015). *Agile everything: An iterative approach can deliver solutions to nearly any problem — whether at work or in life*. PM Network. https://www.pmi.org/learning/library/agile-everything-9690

Fornell, C., & Larcker, D. (1981). Evaluating structural equation models with unobservable variables and

measurement error. *Journal of Marketing Research*, *18*(1), 39–50. https://doi.org/10.2307/3151312

Fowler, M. (2006). *Semantic diffusion*. https://martinfowler.com/bliki/SemanticDiffusion.html

Fowler, M., Beck, K., Brant, J., & Opdyke, W. (1999). *Refactoring: Improving the design of existing code*. Addison Wesley.

Fuchs, C., & Golenhofen, F. J. (2019). Agile for mechatronics and hardware. In C. Fuchs & F. J. Golenhofen (Eds.), *Mastering disruption and innovation in product management: Connecting the dots* (pp. 147–162). Springer International Publishing. https://doi.org/10.1007/978-3-319-93512-6_8

Geiger, J. R. (2020). *Agility measurement for large organizations* [Doctoral dissertation, Air Force Institute of Technology]. https://scholar.afit.edu/cgi/viewcontent.cgi?article=5342&context=etd

Gemino, A., Reich, B. H., & Serrador, P. (2020). Agile, traditional, and hybrid approaches to project success: Is hybrid a poor second choice? *Project Management Journal*, *52*(2), 161–175. https://doi.org/10.1177/8756972820973082

Ghimire, D., & Charters, S. (2022). The impact of agile development practices on project outcomes. *Software*, *1*(3), 265–275. https://doi.org/10.3390/software1030012

Gnizy, I. (2025). When does business agility matter to firm strategic performance. *Journal of Asia Business Studies*, *20*(1), 23–46. https://doi.org/10.1108/JABS-09-2024-0509

Goldman, S. L., Nagel, R. N., & Preiss, K. (1995). *Agile competitors and virtual organizations: Strategies for enriching the customer*. Van Nostrand Reinhold.

Greineder, M., Blohm, I., & Leicht, N. (2020, June). Conceptualizing the agile work organization: A systematic literature review, framework and research agenda. *Proceedings of the 33rd Bled eConference*. https://doi.org/10.18690/978-961-286-362-3.18

Gren, L., Torkar, R., & Feldt, R. (2015). The prospects of a quantitative measurement of agility: A validation study on an agile maturity model. *Journal of Systems and Software*, *107*, 38–49. https://doi.org/10.1016/j.jss.2015.05.008

Gunasekaran, A., Yusuf, Y. Y., Adeleye, E. O., Papadopoulos, T., Kovvuri, D., & Geyi, D. G. (2019). Agile manufacturing: An evolutionary review of practices. *International Journal of Production Research*, *57*(15–16), 5154–5174. https://doi.org/10.1080/00207543.2018.1530478

Haase, M., Jöhnk, J., Lipowsky, S., & Urbach, N. (2017). Der Einfluss des Agilitätsgrads auf den Erfolg von Softwareentwicklungsprojekten unter Berücksichtigung der Unternehmenskultur. In J. M. Leimeister & W. Brenner (Eds.), *Proceedings der 13. Internationalen Tagung Wirtschaftsinformatik* (pp. 454–468). https://www.wi2017.ch/images/wi2017-0356.pdf

Hair, J. F., Ringle, C. M., & Sarstedt, M. (2011). PLS-SEM: Indeed a silver bullet. *Journal of Marketing Theory and Practice*, *19*(2), 139–152. https://doi.org/10.2753/mtp1069-6679190202

Haskin, J. (2022). *Agile software development and successful project completion: A quantitative study* [Doctoral dissertation, University of Phoenix]. https://media.proquest.com/media/hms/PFT/2/SJF2O?_s=2rkJKtwX4t41MXCrBp5W2ubERZc%3D

Henriques, V., & Tanner, M. (2017). A systematic literature review of agile maturity model research. *Interdisciplinary Journal of Information, Knowledge, and Management*, *12*, 053–073. https://doi.org/10.28945/3666

Henseler, J., Ringle, C. M., & Sarstedt, M. (2014). A new criterion for assessing discriminant validity in variance-based structural equation modeling. *Journal of the Academy of Marketing Science*, *43*(1), 115–135. https://doi.org/10.1007/s11747-014-0403-8

Highsmith, J. (2000). *Adaptive software development*. Dorset House Publishing.

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, *34*(9), 120–127. https://doi.org/10.1109/2.947100

Hoda, R., Salleh, N., & Grundy, J. (2018). The rise and evolution of agile software development. *IEEE Software*, *35*(5), 58–63. https://doi.org/10.1109/ms.2018.290111318

Hohl, P., Klünder, J., van Bennekum, A., Lockard, R., Gifford, J., Münch, J., Stupperich, M., & Schneider, K. (2018). Back to the future: Origins and directions of the "Agile Manifesto" – views of the originators. *Journal of Software Engineering Research and Development*, *6*(1). https://doi.org/10.1186/s40411-018-0059-z

Holden, R. J., Boustani, M., & Azar, J. (2021). Agile innovation to transform healthcare: Innovating in complex adaptive systems is an everyday process, not a light bulb event. *BMJ Innovations*, *7*(2), 499–505. https://doi.org/10.1136/bmjinnov-2020-000574

Horlach, B., & Drechsler, A. (2020). It's not easy being agile: Unpacking paradoxes in agile environments. In M. Paasivaara & P. Kruchten (Eds.), *Agile processes in software engineering and extreme programming – Workshops* (pp. 182–189). Springer International Publishing. https://doi.org/10.1007/978-3-030-58858-8_19

Hu, L., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural Equation Modeling: A Multidisciplinary Journal*, *6*(1), 1–55. https://doi.org/10.1080/10705519909540118

Huisman, M. (2000). Imputation of missing item responses: Some simple techniques. *Quality and Quantity*, *34*(4), 331–351. https://doi.org/10.1023/A:1004782230065

Hulland, J. (1999). Use of partial least squares (PLS) in strategic management research: A review of four recent studies. *Strategic Management Journal*, *20*(2), 195–204. https://doi.org/10.1002/(SICI)1097-0266(199902)20:2%3C195::AID-SMJ13%3E3.0.CO;2-7

Hunt, A., & Thomas, D. (1999). *The pragmatic programmer: From journeyman to master*. Addison-Wesley Professional.

Itzik, D., & Roy, G. (2023). Does agile methodology fit all characteristics of software projects? Review and analysis. *Empirical Software Engineering*, *28*(4), 105. https://doi.org/10.1007/s10664-023-10334-7

Jalali, S., Wohlin, C., & Angelis, L. (2014). Investigating the applicability of agility assessment surveys: A case study. *Journal of Systems and Software*, *98*, 172–190. https://doi.org/10.1016/j.jss.2014.08.067

Jeffries, R. (2025, July 29). *Enterprise agility??* https://ronjeffries.com/articles/-w025/y/y/

Jin‐Hai, L., Anderson, A. R., & Harrison, R. T. (2003). The evolution of agile manufacturing. *Business Process Management Journal*, *9*(2), 170–189. https://doi.org/10.1108/14637150310468380

Joiner, B. (2019). Leadership agility for organizational agility. *Journal of Creating Value*, *5*(2), 139–149. https://doi.org/10.1177/2394964319868321

Junker, T., Bakker, A. B., Derks, D., & Molenaar, D. (2022). Agile work practices: Measurement and mechanisms. *European Journal of Work and Organizational Psychology*, *32*(1), 1–22. https://doi.org/10.1080/1359432x.2022.2096439

Junker, T., Bakker, A. B., Gorgievski, M. J., & Derks, D. (2021). Agile work practices and employee proactivity: A multilevel study. *Human Relations*, *75*(12), 2189–2217. https://doi.org/10.1177/00187267211030101

Kakar, A. K. S. (2017). Investigating the motivating potential of software development methods: Insights from a work design perspective. *Pacific Asia Journal of the Association for Information Systems*, 65–96. https://doi.org/10.17705/1pais.09404

Kanwal, F., & Khurram, F. (2022). Understanding agile practices for job satisfaction through job characteristics. *Journal of Professional & Applied Psychology*, *3*(2), 208–217. https://doi.org/10.52053/jpap.v3i2.100

Karekar, H. (2024). *An empirical investigation of the relationship between agile mindset and agile transformation outcomes* [Doctoral dissertation, Swiss School Of Business and Management Geneva]. https://doi.org/10.13140/RG.2.2.36660.05767

Kassab, M., DeFranco, J. F., & Graciano Neto, V. V. (2018). An empirical investigation on the satisfaction levels with the requirements engineering practices: Agile vs. Waterfall. *2018 IEEE International Professional Communication Conference (ProComm)*, 118–124. https://doi.org/10.1109/ProComm.2018.00033

Kiv, S., Heng, S., Kolp, M., & Wautelet, Y. (2017). An intentional perspective on partial agile adoption. *Proceedings of the 12th International Conference on Software Technologies*, 116–127. https://doi.org/10.5220/0006429301160127

Kiv, S., Heng, S., Wautelet, Y., & Kolp, M. (2018). Towards a goal-oriented framework for partial agile adoption. *International Conference on Software Technologies*, *868*, 69–90.

Knutsen, L. Z., Hannay, J. E., & Tanilkan, S. S. (2024). *The problem of the agile bandwagon: Related phenomena in search of conceptual coherence* (SSRN Scholarly Paper No. 4975543). SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4975543

Kuhrmann, M., Tell, P., Hebig, R., Klunder, J. A.-C., Munch, J., Linssen, O., Pfahl, D., Felderer, M., Prause, C., Macdonell, S., Nakatumba-Nabende, J., Raffo, D., Beecham, S., Tuzun, E., Lopez, G., Paez, N., Fontdevila, D., Licorish, S., Kupper, S., … Richardson, I. (2021). What makes agile software development agile. *IEEE Transactions on Software Engineering*, *48*(9), 3523–3539. https://doi.org/10.1109/tse.2021.3099532

Laanti, M., Similä, J., & Abrahamsson, P. (2013). Definitions of agile software development and agility. In F. McCaffery, R. V. O'Connor, & R. Messnarz (Eds.), *Systems, Software and Services Process Improvement* (pp. 247–258). Springer. https://doi.org/10.1007/978-3-642-39179-8_22

Lalić, D., Lalić, B., Delić, M., Gračanin, D., & Stefanović, D. (2022). How project management approach impact project success? From traditional to agile. *International Journal of Managing Projects in Business*, *15*(3), 494–521. https://doi.org/10.1108/ijmpb-04-2021-0108

Lance, C. E., Butts, M. M., & Michels, L. C. (2006). The sources of four commonly reported cutoff criteria. *Organizational Research Methods*, *9*(2), 202–220. https://doi.org/10.1177/1094428105284919

Layik, D. (2021). *Agile vs. Lean: A systematic literature review comparing underlying principles, work-floor practices, and team-level behaviours* [Master's thesis, University of Twente]. https://essay.utwente.nl/fileshare/file/86121/Layik_BA_IMC.pdf

Leppänen, M. (2013). A comparative analysis of agile maturity models. In R. Pooley, J. Coady, C. Schneider, H. Linger, C. Barry, & M. Lang (Eds.), *Information Systems Development: Reflections, challenges and new directions* (pp. 329–343). Springer. https://doi.org/10.1007/978-1-4614-4951-5_27

Li, C.-H. (2016). The performance of ML, DWLS, and ULS estimation with robust corrections in structural equation models with ordinal variables. *Psychological Methods*, *21*(3), 369–387. https://doi.org/10.1037/met0000093

Looks, H., Fangmann, J., Thomaschewski, J., Escalona, M.-J., & Schön, E.-M. (2021). Towards a standardized questionnaire for measuring agility at team level. In P. Gregory, C. Lassenius, X. Wang, & P. Kruchten (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (pp. 71–85). Springer International Publishing. https://doi.org/10.1007/978-3-030-78098-2_5

Madsen, D. Ø. (2020). The evolutionary trajectory of the agile concept viewed from a management fashion perspective. *Social Sciences*, *9*(5), 69. https://doi.org/10.3390/socsci9050069

Magistretti, S., & Trabucchi, D. (2025). Agile-as-a-tool and agile-as-a-culture: A comprehensive review of agile approaches adopting contingency and configuration theories. *Review of Managerial Science*, *19*(1), 223–253. https://doi.org/10.1007/s11846-024-00745-1

Mellor, S., & Balcer, M. (2002). *Executable UML: A foundation for model-driven architecture*. Addison Wesley.

Memeti, A., Huck-Fries, V., Wiesche, M., Thatcher, J. B., & Krcmar, H. (2021). Motivation in IT projects: Investigating the effect of agile practices on team members' intrinsic motivation. *PACIS 2021 Proceedings*. https://aisel.aisnet.org/cgi/viewcontent.cgi?article=1160&context=pacis2021

Meyer, S., Newsome, D., Fuller, T., Newsome, K., & Ghezzi, P. M. (2021). Agility: What it is, how to measure it, and how to use it. *Behavior Analysis in Practice*, *14*(3), 598–607. https://doi.org/10.1007/s40617-020-00465-4

Michaelides, G., Thomson, C., & Wood, S. (2010). Measuring fidelity to extreme programming: A psychometric approach. *Empirical Software Engineering*, *15*(6), 599–617. https://doi.org/10.1007/s10664-010-9130-z

Miller, G. G. (2001). The characteristics of agile software processes. *Proceedings of the 14th International Conference on Technology of Object-Orientated Languages and Systems*, 385–388. https://doi.org/10.1109/TOOLS.2001.10035

Mîndrilă, D. (2010). Maximum likelihood (ML) and diagonally weighted least squares (DWLS) estimation procedures: A comparison of estimation bias with ordinal and multivariate non-normal data. *International Journal for Digital Society*, *1*(1), 60–66. https://doi.org/10.20533/ijds.2040.2570.2010.0010

Miranda, E. (2011). Time boxing planning. *ACM SIGSOFT Software Engineering Notes*, *36*(6), 1–5. https://doi.org/10.1145/2047414.2047428

Moh'd, S., Gregory, P., Barroca, L., & Sharp, H. (2024). Agile human resource management: A systematic mapping study. *German Journal of Human Resource Management*, *38*(4), 345–374. https://doi.org/10.1177/23970022231226316

Mortensen, R., Boje-Nielsen, T., & Jasemian, T. (2007). *Measuring agility—A quantitative survey among IT professionals* [Master's thesis]. https://projekter.aau.dk/projekter/files/61070870/1181300194.pdf

Mushtaq, Z., & Qureshi, M. R. J. (2012). Novel hybrid model: Integrating scrum and XP. *International Journal of Information Technology and Computer Science*, *4*(6), 39–44. https://doi.org/10.5815/ijitcs.2012.06.06

Nájera Catalán, H. E. (2019). Reliability, population classification and weighting in multidimensional poverty measurement: A Monte Carlo study. *Social Indicators Research*, *142*(3), 887–910. https://doi.org/10.1007/s11205-018-1950-z

Nguyen, T., Le, C. V., Nguyen, M., Nguyen, G., Lien, T. T. H., & Nguyen, O. (2024). The organisational impact of agility: A systematic literature review. *Management Review Quarterly*, *75*, 2709–2757. https://doi.org/10.1007/s11301-024-00446-9

Nunnally, J. C. (1978). *Psychometric theory* (2nd ed.). McGraw-Hill.

Osigweh, C. A. B. (1989). Concept fallibility in organizational science. *The Academy of Management Review*, *14*(4), 579–594. https://doi.org/10.2307/258560

Ozkan, N., & Gok, M. S. (2022). Definition synthesis of agility in software development: Comprehensive review of theory to practice. *International Journal of Modern Education and Computer Science*, *14*(3), 26–44.

Palmer, S., & Felsing, J. (2002). *A practical guide to feature-driven development*. Prentice Hall.

Parent, M. C. (2013). Handling item-level missing data: Simpler is just as good. *The Counseling Psychologist*, *41*(4), 568–600. https://doi.org/10.1177/0011000012445176

Petermann, M. K. H., & Zacher, H. (2022). Workforce agility: Development and validation of a multidimensional measure. *Frontiers in Psychology*, *13*, Article 841862. https://doi.org/10.3389/fpsyg.2022.841862

Peterson, R. A., & Kim, Y. (2013). On the relationship between coefficient alpha and composite reliability. *Journal of Applied Psychology*, *98*(1), 194–198. https://doi.org/10.1037/a0030767

Peyre, H., Leplège, A., & Coste, J. (2011). Missing data methods for dealing with missing items in quality of life questionnaires. A comparison by simulation of personal mean score, full information maximum likelihood, multiple imputation, and hot deck techniques applied to the SF-36 in the French 2003 decennial health survey. *Quality of Life Research*, *20*(2), 287–300. https://doi.org/10.1007/s11136-010-9740-3

Pinho, C. R. A., Pinho, M. L. C. A., Deligonul, S., & Çavuşgil, S. T. (2022). The agility construct in the literature: Conceptualization and bibliometric assessment. *Journal of Business Research*, *153*, 517–532. https://doi.org/10.1016/j.jbusres.2022.08.011

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: An agile toolkit*. Addison-Wesley.

Rahman, A., Agrawal, A., Krishna, R., & Menzies, T. (2018). "Doing" agile versus "being" agile. Empirical results from 250+ projects. *Proceedings of the 15th International Conference on Mining Software Repositories*, 311–321.

Rath, S. P., Jain, N. K., Tomer, G., & Singh, A. K. (2025). A systematic literature review of agile software development projects. *Information and Software Technology*, *182*, 107727. https://doi.org/10.1016/j.infsof.2025.107727

Rathor, S., Xia, W., & Batra, D. (2023). Achieving software development agility: Different roles of team,

methodological and process factors. *Information Technology & People*, *37*(2), 835–873. https://doi.org/10.1108/ITP-10-2021-0832

Recker, J., Holten, R., Hummel, M., & Rosenkranz, C. (2017). How agile practices impact customer responsiveness and development success: A field study. *Project Management Journal*, *48*(2), 99–121. https://doi.org/10.1177/875697281704800208

Rhemtulla, M., Brosseau-Liard, P. É., & Savalei, V. (2012). When can categorical variables be treated as continuous? A comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. *Psychological Methods*, *17*(3), 354–373. https://doi.org/10.1037/a0029315

Rietze, S., & Zacher, H. (2022). Relationships between agile work practices and occupational well-being: The role of job demands and resources. *International Journal of Environmental Research and Public Health*, *19*(3), Article 1258. https://doi.org/10.3390/ijerph19031258

Rönkkö, M., & Cho, E. (2020). An updated guideline for assessing discriminant validity. *Organizational Research Methods*, *25*(1), 6–14. https://doi.org/10.1177/1094428120968614

Roth, P. L., Switzer, F. S., & Switzer, D. M. (1999). Missing data in multiple item scales: A Monte Carlo analysis of missing data techniques. *Organizational Research Methods*, *2*(3), 211–232. https://doi.org/10.1177/109442819923001

Rubia, J. M. de la. (2019). Review of the criteria for convergent validity estimated through the Extracted Average Variance. *Psychologia. Avances de La Disciplina*, *13*(2), 25–41. https://doi.org/10.21500/19002386.4119

Rytkönen, J., Mikkonen, T., & Mäkitalo, N. (2025). Empirical evidence on benefits of agile methods: How much we really know? In D. Pfahl, J. Gonzalez Huerta, J. Klünder, & H. Anwar (Eds.), *Product-focused software process improvement* (pp. 317–324). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-78386-9_21

Sandstø, R., & Reme-Ness, C. (2021). Agile practices and impacts on project success. *Journal of Engineering, Project and Production Management*, *11*(3), 255–262. https://doi.org/10.2478/jeppm-2021-0024

Sarstedt, M., Ringle, C. M., & Hair, J. F. (2022). Partial least squares structural equation modeling. In C. Homburg, M. Klarmann, & A. E. Vomberg (Eds.), *Handbook of market research* (pp. 587–632). Springer, Cham. https://doi.org/10.1007/978-3-319-05542-8_15-2

Sartori, G. (1970). Concept misformation in comparative politics. *The American Political Science Review*, *64*(4), 1033–1053. https://doi.org/10.2307/1958356

Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychological Methods*, *7*(2), 147–177. https://doi.org/10.1037/1082-989X.7.2.147

Schirrmacher, A., & Schoop, M. (2018). Agility in information systems – a literature review on terms and definitions. *UK Academy for Information Systems Conference Proceedings 2018*, *25*. https://aisel.aisnet.org/ukais2018/25

Schlein, B. K. (2022, April 20). We need a revised Agile Manifesto. *Agile Insider*. https://medium.com/agileinsider/we-need-a-revised-agile-manifesto-34705ff2cc90

Schwaber, K., & Beedle, M. (2001). *Agile software development with scrum*. Pearson.

Schweigert, T., Vohwinkel, D., Korsaa, M., Nevalainen, R., & Biro, M. (2013). Agile maturity model: Analysing agile maturity characteristics from the SPICE perspective. *Journal of Software: Evolution and Process*, *26*(5), 513–520. https://doi.org/10.1002/smr.1617

Schweizer, K., Troche, S. J., & DiStefano, C. (2019). Scaling the variance of a latent variable while assuring constancy of the model. *Frontiers in Psychology*, *10*, Article 887. https://doi.org/10.3389/fpsyg.2019.00887

Serrador, P., & Pinto, J. K. (2015). Does agile work? A quantitative analysis of agile project success. *International Journal of Project Management*, *33*(5), 1040–1051. https://doi.org/10.1016/j.ijproman.2015.01.006

Sharma, N., & Wadhwa, M. (2015). eXSRUP: Hybrid software development model integrating extreme programing, scrum & rational unified process. *Indonesian Journal of Electrical Engineering*, *16*(2), 377. https://doi.org/10.11591/tijee.v16i2.1627

Silva-Martinez, J., Barth, T. S., Beil, R. J., Holladay, J., & Pawlikowski, G. (2024). *Results from NASA agile teams study*. AIAA SCITECH 2024 Forum (Paper 0408). https://doi.org/10.2514/6.2024-0408

Sjøberg, D. I. K., & Bergersen, G. R. (2023). Construct validity in software engineering. *IEEE Transactions on Software Engineering*, *49*(3), 1374–1396. https://doi.org/10.1109/TSE.2022.3176725

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 P. Abrahamsson, M. Marchesi, & F. Maurer (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (pp. 83–93). Springer. https://doi.org/10.1007/978-3-642-01853-4_11

Solinger, O. N., Heusinkveld, S., & Cornelissen, J. P. (2024). Redefining concepts to build theory: A repertoire for conceptual innovation. *Human Resource Management Review*, *34*(1), 100988. https://doi.org/10.1016/j.hrmr.2023.100988

Stapleton, J. (1997). *DSDM dynamic systems development method: The method in practice: A framework for business centered development*. Pearson Education Limited.

Strode, D. E. (2005). *The agile methods: An analytical comparison of five agile methods and an investigation of their target environment* [Master's thesis, Massey University]. http://hdl.handle.net/10179/515

Syed-Abdullah, S., Holcombe, M., & Gheorge, M. (2006). The impact of an agile methodology on the well being of development teams. *Empirical Software Engineering*, *11*, 143–167. https://doi.org/10.1007/s10664-006-5968-5

Telemaco, U., Alencar, P., Cowan, D., & Oliveira, T. (2022). *Agile assessment methods: Current state of the art* (arXiv:2212.10808). arXiv. https://doi.org/10.48550/arXiv.2212.10808

Thomas, D. (2014, March). *Agile is dead (long live agility)*. PragDave. https://pragdave.me/thoughts/active/2014-03-04-time-to-kill-agile.html

Tripp, J., Rienemschneider, C., & Thatcher, J. (2016). Job satisfaction in agile development teams: Agile development as work redesign. *Journal of the Association for Information Systems*, *17*(4), 267–307. https://doi.org/10.17705/1jais.00426

Tripp, J., Saltz, J., & Turk, D. (2018). Thoughts on current and future research on agile and lean: Ensuring relevance and rigor. *Proceedings of the 51st Hawaii International Conference on System Sciences*, 5465–5472. https://doi.org/10.24251/HICSS.2018.681

Tuncel, D., Körner, C., & Plösch, R. (2020). Comparison of agile maturity models: Reflecting the real needs. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 51–58. https://doi.org/10.1109/SEAA51224.2020.00019

Tuncel, D., Körner, C., & Plösch, R. (2022). Questionnaire development for a scientifically founded agile assessment model. *Joint Conference of the 31st International Workshop on Software Measurement (IWSM) and the 16th International Conference on Software Process and Product Measurement (MENSURA)*, *3272*, 1–16.

Umer, M., Nazır, T., Muhammad, N., Maqsoom, A., Nawab, S., Fatima, S. T., Shafi, K., & Butt, F. S. (2021). Impact of agile management on project performance: Evidence from I.T sector of Pakistan. *PLOS ONE*, *16*(4), Article e0249311. https://doi.org/10.1371/journal.pone.0249311

Uraon, R. S., Chauhan, A., Bharati, R., & Sahu, K. (2023). Do agile work practices impact team performance through project commitment? Evidence from the information technology industry. *International Journal of Productivity and Performance Management*, *73*(4), 1212–1234. https://doi.org/10.1108/ijppm-03-2023-0114

Vallon, R., da Silva Estácio, B. J., Prikladnicki, R., & Grechenig, T. (2018). Systematic literature review on agile practices in global software development. *Information and Software Technology*, *96*, 161–180. https://doi.org/10.1016/j.infsof.2017.12.004

Vasylieva, K., Kuhrmann, M., Xavier, M. K., & Klünder, J. (2023). How agile are you? Discussing maturity levels of agile maturity models. *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 270–277. https://doi.org/10.1109/SEAA60479.2023.00049

Vihovde, E. H., & Meng, Q. (2024). Test-driven development: Ensuring code quality in integration with CI/CD. *Proceedings of 2024 8th International Conference on Management Engineering, Software Engineering and Service Sciences (ICMSS)*, 8–11. https://doi.org/10.1109/ICMSS61211.2024.00009

Visconti, M., & Cook, C. R. (2004). An ideal process model for agile methods. In F. Bomarius & H. Iida (Eds.), *Product Focused Software Process Improvement* (Vol. 3009, pp. 431–441). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-24659-6_31

Walter, A.-T. (2021). Organizational agility: Ill-defined and somewhat confusing? A systematic literature review and conceptualization. *Management Review Quarterly*, *71*(2), 343–391. https://doi.org/10.1007/s11301-020-00186-6

Welch, C., Rumyantseva, M., & Hewerdine, L. J. (2016). Using case research to reconstruct concepts: A methodology and illustration. *Organizational Research Methods*, *19*(1), 111–130. https://doi.org/10.1177/1094428115596435

Wendler, R. (2013). The structure and components of agility – A multi-perspective view. *Informatyka Ekonomiczna*, *2*(28), 148–169.

Whiteley, A., Pollack, J., & Matouš, P. (2021). The origins of agile and iterative methods. *The Journal of Modern Project Management*, *8*(3), 20–29. https://doi.org/10.19255/jmpm02502

Wieland, A., Durach, C. F., Kembro, J., & Treiblmaier, H. (2017). Statistical and judgmental criteria for scale purification. *Supply Chain Management: An International Journal*, *22*(4), 321–328. https://doi.org/10.1108/scm-07-2016-0230

Williams, L., Krebs, W., & Layman, L. (2004). Extreme programming valuation framework for object-oriented languages version 1.3: (Technical Report TR-2004-11). *North Carolina State University, Department of Computer Science*. https://ncsu.edu

Wood, S., Michaelides, G., & Thomson, C. J. (2013). Successful extreme programming: Fidelity to the methodology or good teamworking? *Information & Software Technology*, *55*(4), 660–672. https://doi.org/10.1016/j.infsof.2012.10.002

Yang, N., Wang, X., Zhang, Z., Siemon, D., & Hyrynsalmi, S. (2025). Core theories in agile software development. In S. Peter, M. Kropp, A. Aguiar, C. Anslow, M. I. Lunesu, & A. Pinna (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (pp. 3–18). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-94544-1_1

Yauch, C. A. (2011). Measuring agility as a performance outcome. *Journal of Manufacturing Technology Management*, *22*(3), 384–404. https://doi.org/10.1108/17410381111112738

Yusuf, Y. Y., Sarhadi, M., & Gunasekaran, A. (1999). Agile manufacturing: The drivers, concepts and attributes. *International Journal of Production Economics*, *62*(1), 33–43. https://doi.org/10.1016/S0925-5273(98)00219-9

APPENDIX A
QUESTIONNAIRE TO ASSESS AGILITY IN SOFTWARE ENGINEERING (QASE)

N°	Factor (items)	Item Wording	Model						
			M1	M2	M3	M4	M5	M6	M7
1	Pair Programming/Modeling								
	PP1	When software is being developed, developers discuss their approach with each other.	*	*	*	*	*		
	PP2[4*]	When software is being developed, two programmers concentrate on the code being written.	*	*	*	*	*		
	PP3[4+]	Your design, model, or code is created by two people working together at a single computer.	*	*	*				
	PP4	**In your team, people like to program in pairs.**				*	*	*	*
	PP5	**The knowledge of how a component was developed is shared by at least two persons in the team.**				*	*	*	*
	PP6	**Experience/knowledge is exchanged and shared between the developers during programming.**				*	*	*	*
2	Simple Design								
	SD1	**To keep the code/model simple is of high importance in your team.**	*	*	*				*
	SD2	You have quality metrics to ensure the simplicity of the code/model, and you stick with them even in crunch time.	*						
	SD3	**When new software is being developed, you focus on keeping the solution as simple as possible.**	*	*	*				*
	SD4	**You always manage to keep your code/model as simple as possible.**	*	*	*				*
3	Mission-Driven Development								
	MDD1	**The business goals/drivers of your project are clearly defined.**	*	*	*				*
	MDD2	**You are aware of the business goals/drivers of your project.**	*	*	*				*
	MDD3	Even when you are operating on a technical level, you still keep the business drivers of your project in mind.	*						
	MDD4	Your policy is to only develop what has direct business value.	*						
	MDD5	**The activities in each development cycle can be justified against the overall project mission.**	*	*	*				*
4	Short Release Cycles								
	SR1	Your development cycle is short enough to manage (customer) feedback with little effort.	*	*	*	*	*		
	SR2	Your development cycle is short and gives you the feeling not to be busy with the same task for too long.	*	*	*				
	SR3	**Your development cycle is short enough to keep you focused on short-term goals.**	*	*	*	*	*	*	*
	SR4	You regularly get early (customer) feedback on the team's achievement, as your development cycle is short.	*	*		*	*		
	SR5	**Your development cycle is short enough to handle the impact of (ever-)changing requirements well.**				*	*	*	*
	SR6	**Your development cycle is short enough to adapt well to unexpected capacity bottlenecks.**				*	*	*	*
	SR7	The consequences of implementing the wrong requirements are minimal because your development cycle is short enough to identify them early.				*	*		
5	Continuous Project Monitoring								
	CPM1[4*]	**Your team utilizes visual indicators (charts, graphs, etc.) of how well you are progressing during a work cycle.**	*	*	*				*
	CPM2[4+]	**You use visual tools that allow team members to easily tell if the work is being completed on schedule.**	*	*	*				*
	CPM3[4+]	**You plot your work completed against the work planned on a chart.**	*	*	*				*
6	Competence Coverage								
	COC1	**Your team is well positioned as you have the appropriate experience/expertise to deliver the required customer value.**	*	*	*				*
	COC2	**For any technical issue or question, there is at least one person in the team with the relevant experience/competency.**	*	*	*				*
	COC3	**You always manage to communicate with a subject matter expert in your team when required.**	*	*	*				*
7	Team-Based Planning								
	TBP1[1+]	**The planning and estimation for your project is performed in collaboration with the entire development team.**	*	*	*				*
	TBP2[3*]	**All concerns from team members about reaching the iteration goals were considered.**	*	*	*				*
	TBP3[2+]	During the project planning, you feel that your participation is important.	*	*					
	TBP4[1+]	**The scope of work for a development cycle is decided by the complete team.**	*	*	*				*
8	Feature-Driven Development								
	FDD1	The complete team works with the aim of developing a (or a set of) feature(s) instead of executing individual tasks.	*	*	*	*	*		
	FDD2	**You are aware of the contribution of your tasks to the fulfillment of the customer feature to be delivered at the end of the development cycle.**	*	*		*	*	*	*
	FDD3	The success of a development cycle is measured by the number of implemented software features rather than the number of tasks completed.	*	*	*				
	FDD4	Your project plan is defined in such a way that all the activities needed to develop a feature are performed in a single development cycle.	*	*	*				
	FDD5	**The progress in the project is measured by the number of features (representing direct added value for the customer) that have been completed.**				*	*	*	*
	FDD6	**The team is aware of the added value that the to-be-implemented features represent for the customer.**				*	*	*	*
	FDD7	Your work results in (or contributes to) a tangible outcome that is useful in the eyes of the customer.				*	*		
9	Shared Product Understanding								
	SPU1	At the beginning of the project, you created an ordered list of everything that is known to be needed in the product.	*	*					
	SPU2	The overview of all currently known product features to be developed is well known to everyone in the team.	*	*					
	SPU3	**At the beginning of the project, the product vision and the major product characteristics are defined and shared with the complete team.**	*	*	*				*
	SPU4	**The complete development team is aligned toward the product vision.**	*	*	*				*
	SPU5	**You can see the complete picture and make informed decisions, as you have an overview of all currently known product features to be developed.**	*	*	*				*

Note: The boldfaced items represent the final version of the QASE. * indicates takeover without modification, + indicates takeover with modification, 1 indicates takeover from Looks et al. (2021), 2 indicates takeover from Gren et al (2015), 3 indicates takeover from So & Scholl (2009), 4 indicates takeover from Tripp et al. (2016), 5 indicates takeover from Williams et al. (2004), 6 indicates takeover from Wood et al. (2013).

APPENDIX A
QASE: QUESTIONNAIRE TO ASSESS AGILITY IN SOFTWARE ENGINEERING (QASE) (CONTINUED)

N°	Factor (items)	Item Wording	Model						
			M1	M2	M3	M4	M5	M6	M7
10	Retrospective								
	RETRO1	At the beginning of each work cycle, the team seeks to improve its performance based on the retrospective/lessons learned of the previous work cycle.	*	*					
	RETRO2[1+]	You don't make the same mistakes because in regular retrospectives, the project's approach is reflected, with the aim of improvement.	*	*					
	RETRO3	**During the retrospective, you feel that your participation is important.**	*	*	*				*
	RETRO4[1*]	**Insights gained from retrospectives are turned into concrete improvement measures.**	*	*	*				*
	RETRO5[3+]	**The retrospectives helped you become aware of what you did well in past iterations.**	*	*	*				*
11	Collective Ownership								
	COW1[1*]	**The source code is considered collective property by the entire team.**	*	*	*	*	*	*	*
	COW2	You feel responsible for the quality of not only your work but also the complete software.	*	*		*	*		
	COW3	**Any developer has the ability to change any line of code to add functionalities, fix bugs, improve designs, or refactor as needed.**	*	*	*				*
	COW4	**Each team member has the responsibility and authority to maintain every part of the code.**	*	*	*	*	*		*
	COW5	The entire team is accountable for the quality of the overall code.				*	*	*	
	COW6	The quality (good or poor) of the overall code is a team achievement.				*	*	*	
	COW7	Each team member is welcome to contribute new ideas/improvements to any part of the overall code.				*	*		
12	Function Prioritization								
	PRIO1	The functions to be implemented are prioritized by the customer in collaboration with the team.	*						
	PRIO2	The complete development team actively participates in the prioritization of the functions to be developed.	*						
	PRIO3	**You analyze and prioritize the functions to be developed in order to handle potential risks.**	*	*	*				*
	PRIO4	**You analyze and prioritize the functions to be developed in order to work in an efficient way.**	*	*	*				*
	PRIO5	**You analyze and prioritize the functions to be developed in order to bring early value to your customer.**	*	*	*				*
13	Role Homogeneity								
	ROH1	**There is no role separation in your team, and therefore, you are interchangeable in your activities.**	*	*	*	*	*	*	*
	ROH2	In your team, you do not differentiate between architect, tester, programmer, integrator, etc., as everyone does everything.	*	*	*				
	ROH3	In your team, each member is responsible for the complete development cycle of an entire piece of work.	*	*	*	*	*		
	ROH4	**In your team, any activity can be assigned to any team member, regardless of the role required to perform that activity.**				*	*	*	*
	ROH5	There is no time dependency between the team members in fulfilling their respective tasks in the project.				*	*		
	ROH6	**Each team member is a generalist rather than a domain specialist in relation to his/her role in the project.**				*	*	*	*
14	Decentralized Decision-Making								
	DDM1[6+]	Your team is designed to let every team member participate in decision-making.	*	*					
	DDM2[1*]	**Decisions regarding the execution of their own work can be made by the team without the involvement of a managing authority.**	*	*	*				*
	DDM3[6*]	As a member of the team, you have a real say in how the team carries out its work.	*	*					
	DDM4	**All decisions regarding the execution of the project are up to the development team and are made within the team.**	*	*	*				*
	DDM5[2+]	**The development team has the authority to make decisions without the need to refer back to any manager.**	*	*	*				*
15	Sustainable Work Pace								
	SWP1	**You are able to complete your work in your usual working hours and take a break whenever you feel the need to.**	*	*	*				*
	SWP2	**You are comfortable with your usual workload in the project.**	*	*	*				*
	SWP3	**Your workload allows you to have a good life-domain balance.**	*	*	*				*
16	Refactoring								
	REF1[5+]	**You frequently spend time cleaning up code without changing functionality.**	*	*	*				*
	REF2[4*]	**You frequently spend time improving the design of previously written code.**	*	*	*				*
	REF3[4*]	Each member of the team attempts to improve the structure of the code when making a change.	*	*					
	REF4[2+]	**You are willing to rework the design/code/model to keep it clean, even if it requires some rework of already completed work products.**	*	*	*				*
17	Management Involvement								
	MNG1	**Your software is developed in interaction with the management.**	*	*	*				*
	MNG2	The developers communicate directly with the management without any hurdles or through a management contact person.	*	*					
	MNG3	**Your management continuously strives to provide the support the team needs to be successful in the project.**	*	*	*				*
	MNG4	**Your management is aware of what is going on in your project.**	*	*	*				*

Note: The boldfaced items represent the final version of the QASE. * indicates takeover without modification, + indicates takeover with modification, 1 indicates takeover from Looks et al. (2021), 2 indicates takeover from Gren et al (2015), 3 indicates takeover from So & Scholl (2009), 4 indicates takeover from Tripp et al. (2016), 5 indicates takeover from Williams et al. (2004), 6 indicates takeover from Wood et al. (2013).

APPENDIX A
QASE: QUESTIONNAIRE TO ASSESS AGILITY IN SOFTWARE ENGINEERING (QASE) (CONTINUED)

N°	Factor (items)	Item Wording	Model						
			M1	M2	M3	M4	M5	M6	M7
18	Change and Configuration Management								
	CCM1	**Your software integration is accomplished without significant issues, as every change in your software is properly documented.**	*	*	*				*
	CCM2	**If a problem is identified in a certain version or branch of your software, it is easy for you to identify other impacted versions.**	*	*	*				*
	CCM3	You always manage to deliver the right software version with all the intended functions to your customer without issues.	*	*					
	CCM4	You are able to easily track and understand the changes introduced in your software without significant issues when required.	*	*					
	CCM5	**You take the necessary time to document the changes you made to make them traceable, even in crunch time.**	*	*	*				*
	CCM6	Each team member can easily continue working on someone else's work package, as you thoroughly document the changes you make to your development items.	*	*					
19	Standards-Based Development								
	SBD1	**The conventions/templates/standards to be used in your project are clearly defined and easy to understand.**	*	*	*				*
	SBD2	**The conventions/templates/standards to be used in your project are well known to everyone in the team.**	*	*	*				*
	SBD3	**You quickly get familiar with your colleague's development items, as you are using the same conventions/templates.**	*	*	*				*
	SBD4	For each team member, any part of the code is easily readable, maintainable, and extensible, as you are using a common convention.	*						
20	Documentation Commitment								
	DCM1	**The preparation of a "user manual" for your software is an integral part of your development cycle.**	*	*	*				*
	DCM2	**You have the necessary time to document your software well for the intended user.**	*	*	*				*
	DCM3	**You take the necessary time to prepare the "user manual" (document your software for the intended user), even in crunch time.**	*	*	*				*
21	Automation								
	AUTO1	**All boring/repetitive processes are automated in your project, and you can optimally invest your time in the remaining activities.**	*	*	*				*
	AUTO2	**You highly value automation in your project; therefore, whenever your team sees the possibility to automate something, you do it.**	*	*	*				*
	AUTO3	**The level of automation in your project corresponds to the actual state of technology.**	*	*	*				*
	AUTO4	You can repeat all your tests (on all test levels) at any time, as they are completely automated.	*	*					
22	Iterative & Incremental Development								
	IID1	The scope of the next development cycle depends on the feedback of the customer on the current one.	*	*					
	IID2	Each development cycle aims either to add new functionalities to the software or to refine the previous one based on customer feedback.	*	*	*				
	IID3	Your complete development plan is divided into smaller pieces, and each piece undergoes a complete software development cycle.	*	*	*	*			
	IID4	Each software development cycle is completed with the delivery of the working product to the customer.	*	*	*				
	IID5	Your development process is suitable for starting a project without a full definition of the requirements at the beginning of the project.				*			
	IID6	In your project, you have as many milestones as you have development cycles.				*			
	IID7	**If the customer stopped the project at the end of any development cycle, they would still have a tangible working and usable product.**				*	*	*	*
	IID8	**Your development process consists of repetitive development cycles, and each cycle gives you the feeling of having achieved and completed something tangible.**				*	*	*	*
	IID9	**Your development process consists of repetitive development cycles, and each cycle ends with software that exists and functions independently of further implementations.**				*	*	*	*
	IID10	You split and plan your release/sprint in such a way that each delivery is an executable new functional view of the overall product.				*	*		
23	Active User Involvement								
	AUI1	Your software is developed in interaction with the customer.	*	*					
	AUI2	**The customer is involved in each phase of the development cycle.**	*	*	*				*
	AUI3	**Your customer is aware of what is going on in your project. There is complete transparency.**	*	*	*				*
	AUI4	**Timely decisions can be made on a daily basis with the customer.**	*	*	*				*
	AUI5	You directly communicate with the customer rather than resorting to lengthy documents.	*	*					
24	Change Tolerance								
	CHT1	Changed requirements are seen as an added value of the product for the customer and not as an additional workload.	*	*					
	CHT2[1+]	**You are prepared for requirement changes during the course of the project and are not surprised by them.**	*	*	*				*
	CHT3[1*]	**Already proposed changes in the requirements are allowed to be adapted again by the customer during the project.**	*	*	*				*
	CHT4[2+]	**You are willing to undergo a requirement change even if it requires some reworking of already completed work products.**	*	*	*				*

Note: The boldfaced items represent the final version of the QASE. * indicates takeover without modification, + indicates takeover with modification, 1 indicates takeover from Looks et al. (2021), 2 indicates takeover from Gren et al (2015), 3 indicates takeover from So & Scholl (2009), 4 indicates takeover from Tripp et al. (2016), 5 indicates takeover from Williams et al. (2004), 6 indicates takeover from Wood et al. (2013).

APPENDIX A
QASE: QUESTIONNAIRE TO ASSESS AGILITY IN SOFTWARE ENGINEERING (QASE) (CONTINUED)

N°	Factor (items)	Item Wording	Model						
			M1	M2	M3	M4	M5	M6	M7
25	Timeboxing								
	TBX1[3+]	When the scope of a release/sprint could not be implemented due to constraints, the team reduces the scope rather than delays the deadline.	*	*	*	*			
	TBX2	The planned duration of a development cycle is a fixed parameter in your project and is not exceeded.	*	*	*				
	TBX3	You do not make use of overtime or additional personnel resources to complete the planned scope of a development cycle.	*						
	TBX4	When the time planned for the development cycle is over, you close the cycle independently of whether all the goals have been reached.	*	*					
	TBX5	The scope of a development cycle depends on the duration of the development cycle and not the opposite.	*	*	*	*			
	TBX6	**Your team makes the best use of given time for each release/sprint by working focused and concentrated on what is important.**				*			*
	TBX7	You do not adjust your time to the tasks, but try to get as much done as possible in the given time.				*			
	TBX8	**The usage of timeframes in your project helps you to work focused and to concentrate on what is important.**				*			*
	TBX9	**The usage of timeframes in your project helps you to treasure your time and make the best use of it.**				*			*
	TBX10	In your project, you strive not to use the time allocated to a defined activity for anything other than that activity.				*			
	TBX11	You set a timeframe for each activity and do not deviate from the topic while working on it.				*			
26	Regular Meeting								
	RM1	Your regular meeting is the place where each team member can find instant help.	*	*					
	RM2	**You feel encouraged and supported after your regular meeting.**	*	*	*				*
	RM3	Owing to your frequent team meetings, you are aware of what the other team members are currently working on.	*	*					
	RM4	**Your regular meetings are a key element for the success of your project.**	*	*	*				*
	RM5	**The information shared during your regular meeting is important for your daily work.**	*	*	*				*
27	Open Workspace								
	OPW1	Your workplace is structured in such a way that you will immediately notice if a colleague is out of the office.	*	*					
	OPW2	You manage to communicate with your team colleagues in a timely manner whenever required.	*	*	*	*			
	OPW3	You are communicating more face-to-face or via telephone than via e-mail.	*	*	*				
	OPW4	If you have a question, you can directly go to your colleague; you don't need to schedule a meeting.	*	*	*				
	OPW7	If you have a question, you can contact your colleague immediately; you don't need to schedule a meeting.				*			
	OPW8	You have informal exchanges with your teammate several times a day.				*			
	OPW9	**You would describe the team working environment as hierarchy-free and friendly.**				*			*
	OPW10	**You would describe your work environment as welcoming and supportive.**				*			*
	OPW11	**Anyone in the team can spontaneously share his/her ideas or questions with the rest of the team at any time.**				*			*
28	Review Practice								
	REW1	**You have enough time to properly perform the planned reviews.**	*	*	*				*
	REW2	The time you spend performing reviews in your project is well invested time.	*	*					
	REW3	**You see a benefit in the reviews performed on your development items.**	*	*	*				*
	REW4	**You make use of reviews in your team, and they are an effective instrument to identify issues early.**	*	*	*				*
29	Test-Driven Development								
	TDD1	You strive to gain early feedback on the written code; therefore, you do not have untested software at any point in time.	*	*					
	TDD2	**Tests are written before the code that will make them pass.**	*	*	*				*
	TDD3	**The implementation of the code proceeds step by step until all tests pass.**	*	*	*	*			*
	TDD4	**Unit testing is an integral part of your code development.**	*	*	*				*
	TDD5	Team members synchronize their part of the code continuously to ensure that the complete code always remains correct.				*			
	TDD6	Testing is not considered a phase in your development process, but is done continuously.				*			
	TDD7	You do not develop in order to test at the end, but you test in order to know how to develop.				*			
	TDD8	You strive to get quick feedback on each change you make in your software and its impact on the complete software.				*			
	TDD9	In your team, testing has the highest priority in the development process.				*			
30	Continuous Integration								
	CI1[4+]	Programmers are responsible for personally running a set of unit tests until they all run successfully before delivering changes.	*	*					
	CI2	Programmers are responsible for personally integrating their changes and solving the occurring merge issues.	*	*					
	CI3	You are directly notified of any change, merge, or new version of the software.	*	*					
	CI4	**Each code delivery or "checking in" automatically triggers a test process.**	*	*	*				*
	CI5	**For each change introduced in your software, a set of regression tests is performed before delivering it to the main branch.**	*	*	*				*
	CI6	You directly test each change you made in your software to get quick feedback on its impact on the complete software.	*	*					
	CI7	**Your development environment provides you with the possibility to get instant feedback on any change in your software.**	*	*	*				*

CI8[3+]	Each team member works on the most recent version of code available, as you always deliver your changes as quickly as possible.	*	*

Note: The boldfaced items represent the final version of the QASE. + indicates takeover with modification, 3 indicates takeover from So & Scholl (2009), 4 indicates takeover from Tripp et al. (2016).