Socio-Intentional Framework for Agile Methods Tailoring

by **Soreangsey KIV**

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in Economics and Management Sciences of the Université catholique de Louvain

Examination Committee:

President of the jury: Prof. Jean Vanderdonckt – UCLouvain
Advisor: Prof. Manuel Kolp – UCLouvain
Advisor: Prof. Yves Wautelet – KU Leuven
Examiner: Prof. Samedi Heng – ULiège
Examiner: Prof. Stephan Poelmans – KU Leuven
Examiner: Prof. Saïd Assar – Institut Mines-Télécom (IMT)
Examiner: Prof. Naji Habra – UNamur

October 2021

Acknowledgements

Throughout my PhD journey, I have received enormous support and assistance in different ways. This dissertation would not have been possible without the precious help and encouragement of many people to whom I would like to express my gratitude hereafter.

First and foremost, I would like to express my sincere gratitude to my advisors, **Professor Manuel Kolp** and **Professor Yves Wautelet**, for their continuous support, advice, and encouragement. I am so grateful that they have confidence in me and gave me the freedom to choose the topic of my interest. At the same time, they have provided perfect guidance that allowed me to achieve my goals. I could not have imagined having better advisors.

Prof. Manuel Kolp, I am sincerely grateful for the opportunity you offered me, the teaching and research assistant at the Louvain School of Management. Teaching and research are truly the things I love most in a job. In addition, you are the kindest advisor anyone could have wished for. In the entire journey of my PhD, you have always made time to advise and guide me with patience. With your extensive knowledge and experience, you have helped me accomplish both my assistant and research works in a much simpler way. I am so thankful that you have given me such a stress-free work environment.

Prof. Yves Wautelet, you are my role model and my inspiration. I am sincerely thankful for your invaluable advice, continuous support, patience, and motivation during my PhD. Every discussion with you gave me insightful knowledge and it has tremendously enhanced my ability in conducting research and writing scientific content. In every obstacle that I have encountered during this PhD, your immense knowledge, plentiful experience, and encouragement have helped me get back on track and made me try even harder to achieve my goals. It would not have been possible for me to achieve this PhD without your help and your guidance.

I would like to thank the other members of my PhD committee: Prof. Jean Vanderdonckt (UCLouvain), Prof. Samedi Heng (ULiège), Prof. Stephan Poelmans (KU Leuven), Prof. Saïd Assar (Institut Mines-Télécom), and Prof. Naji Habra (UNamur) for accepting to participate in the jury, for carefully reviewing my thesis and for their stimulating questions and insightful feedback.

Many thanks go to my former coworker, Prof. Samedi Heng, for all his guidance and his assistance throughout my PhD. Bong Samedi, you are not only a great coworker, but also an incredible mentor. I truly appreciate that you keep pushing me outside of my comfort zone, and encouraging me to be the best version of myself. I am so lucky that I have had the chance to work with

you. I believe our discussions have largely improved my research experience and allowed me to make the contributions presented in this thesis. I deeply appreciate your help in work, research, and daily life.

In addition, I would like to thank all my friends and colleagues of Louvain Research Institute in Management and Organization (LouRIM) for their help and the nice work environment. Many sincere thanks go to Prof. Marco Saerens, Thanh-Diane Nguyen, Mathieu Zen, Iyad Khadam, Nesrine Mezhoudi, Sylvain Courtain, Ghazaleh Aghakhani, Pierre Leleux, Mehdi Ousmer, Nicolas Burny, and Vu Nguyen Huynh Anh.

My sincere gratitude also goes to Mrs. Sylvie Baudine, Mrs. Sandrine Delhaye, Mrs. Jasmine De Wulf, and Mrs. Heike Rämer for helping and facilitating the administrative works. Mrs. Sylvie in particular, you are so kind, helpful, and patient. I could not thank you enough for reviewing many of my works.

I am wholeheartedly grateful to my best friend Sovann En who has helped me go through a lot of obstacle for more than 10 years. Sovann, I would not be able to come this far without your continuous help, support, advice, motivation, care, and love. Every bit of your effort is sincerely appreciated.

I would like to offer my special thanks to my dear friend Dona Valy for his care, encouragement, and sympathetic ear. Bong Dona, I admire your positive outlook and your ability to make me laugh despite the situation. I would also like to extend my appreciation to all the Cambodian families and friends in Belgium. My sincere appreciation goes to Samnang Nary, Rachana Ro, Hasika Meth, Kuchvichea Kan, Sokny Ly, Pinnara Ket, Elen Morm, Kannary Keth, Kimnenh Taing, Makara Long, Vathna Lay, and Ratha Siv. I will never forget the wonderful and fun memories we have shared. The times with you were my happy distractions to rest my mind outside of my research.

Last but not least, my very special thanks go to my family, Sokha Kiv, Samay Kieng, Sonissay Kiv, Sodany Kiv, Sodana Kiv, Theary Long, Julien Lesouëf, and Chanhan Hy for their unconditional support, care, and love. My dear Mom and Dad, this thesis is dedicated especially to you. I could not describe how grateful I am for all your hard works and sacrifices to provide a warm, happy family, and every possibility in my life. You are the ultimate reason that I could have come this far. I would also like to thank my brothers and sisters for their constant love and support. I am so blessed to have a family I can always count on when times are rough. I love you all so much.

Soreangsey Kiv

Abstract

Agile is nowadays one of the most-used software development approaches. To gain the full benefit from adopting agile methods, software development teams can choose to adopt agile methods on a custom-basis depending on the context and defined criteria. Over the years, a vast amount of empirical studies aiming to share adoption experiences and introduce the possibility to customize agile methods has been published. Two key aspects in these studies are the goal-orientation and the social aspect. Intuitively, a team adopts agile methods because there are some things it wants to achieve. Once the team identifies the right practices to achieve its goals, the adoption result highly depends on the social factors, including the individuals, their interactions and collaborations.

In this thesis, we propose a socio-intentional framework for tailoring agile methods that allows practitioners to analyze agile practices and to define the right strategies for adopting them. The objective of our framework is to help practitioners decide which agile practices to adopt based on what they want to achieve and identify beforehand how team members should work together to successfully adopt them.

To achieve this objective, we proceeded through four main contributions. In the first contribution, we conducted a Systematic Literature Review (SLR) to extract the motivation behind each agile method's adoption and compared it with the precepts given in the Agile Manifesto. The results show that the Agile Manifesto is highly relevant to these motivations. We conclude that the Agile Manifesto can be used as a criterion for practice selection. In the second contribution, we made knowledge about agile practices adoption reusable in a systematic manner by building an ontology. We started by conducting another SLR to gather knowledge. Based on this, we have built and theoretically validated our ontology. Finally, we conducted a survey with agile experts for empirical validation. In the third contribution, we have built a user-friendly tool that allows practitioners to get the information from the ontology easily. In the last contribution, we proposed a methodology to analyze agile practices on two different levels. The tactical level allows practitioners to decide what agile practice they should adopt based on what they want to achieve and to check the suitability of the selected practice(s) to the team's situations. The operational level allows practitioners to identify beforehand the vulnerabilities in a practice adoption and define how to avoid or solve them. For each level, we showed how to get the knowledge from our tool and to use modeling techniques to ease the analyzing process. Finally, we showed how our framework can be applied within a real software development team.

Table of contents

List of figures

List of tables

Part I

Introduction

Chapter 1

Introduction

This chapter introduces the whole thesis. It is organized as follows. Section 1.1 overviews the research context and the objectives of the thesis. Section 1.2 describes the research design we follow to achieve the objectives. Finally, Section 1.3 provides the reading road-map of this thesis.

1.1 Research Context

In traditional software development, the software is developed in a sequential process. It typically starts with requirements specification and ends with product delivery. In user-intensive software development, to effectively deal with quality expectations, change and risk management, and ensure that all the requirements are satisfied, activities such as requirement gathering, design and development, and testing become iterative and incremental. Among many approaches introduced in the last two decades, there was the emergence of agile methods [66] to offer alternatives to traditional approaches. The new agile methodologies include eXtreme Programming (XP) [17], Feature-Driven Development (FDD) [113], Dynamic Systems Development Method (DSDM) [145], Crystal family [38], Scrum [136], etc. Each of these methodologies was proposed with its own set of values, principles, and practices for practitioners to follow. However, as there is no method that can be a one-size-fits-all, simply choosing a particular agile methodology and following every rule is not an efficient solution. Ideally, to avoid wasting efforts and resources on irrelevant things, the software development team should adopt agile methods differently according to context and criteria. For these reasons, agile methods tailoring has gained a lot of interest in the agile community and it has always been actively studied.

One among many aspects related to agile methods tailoring that have been actively studied is the notion of *goal* [97, 56, 10, 139]. Intuitively, when software development teams want to partially adopt agile methods — either from one or a combination of methodologies — they should have in mind the objectives they want to achieve after the adoption. According to John and Deborah [156], there are different motivations behind agile methods adoption. Campanelli and Parreiras [32] conducted a Systematic Literature Review (SLR) of agile methods tailoring. Their results show that 42.9% of the papers use business

goals as a criterion for agile practice selection. Alongside the business goals, there is another group of researches that shows a strong influence of the Agile Manifesto in choosing agile practices [100, 83, 9, 94]. According to Madi et al. [100], knowing agile values is the key to follow the best set of practices as agile values are fundamental. However, in practice, we have observed that many software development teams decided to adopt agile methods without dedicating any effort to understanding any agile value or principle [12, 16, 26, 48, 140]. Among 18 criteria for agile methods tailoring identified by the same SLR [32], none of them is either an agile value or principle. As the relationship between the Agile Manifesto and the practice is still very skeptical, a formal validation for their relationship thus needs to be done.

Another aspect that is highly relevant and important to agile methods tailoring is *social*. In the software development process, especially agile, the individuals, the interactions and collaborations are the main factors in a successful agile methods adoption. Van Kelle et al. [159] assessed 40 projects to identify the success factors in an agile project. The results show that success (or failure) cannot be determined by the project size, but rather the social factors. Another similar finding from Eckstein [50] shows that most projects do not fail due to technology, but rather social, organizational problems and a lack of effective communication. Even though the social plays an important role in the success of agile methods adoption, there are not many studies on the agile methods tailoring that focus on this aspect. There are thus many problems and limitations within these approaches that need to be addressed.

In any tailoring approach, knowledge is really important for the process to analyze agile methods or practices. For instance, to select the right practice that allows achieving targeted goals, we need to know the benefits of each practice. After years of experience with agile methods, such information can be vastly found in both academia and industrial knowledge bases. Many empirical studies of agile methods adoption have been published to share the experiences about agile practices/methods adoption, about customizing agile practices for a given situation, and how to enable team members to collaborate efficiently, etc. Even though this knowledge is very useful, it is however time-consuming to collect and classify manually. It is the reason why many teams decide to adopt a particular agile method or practice without considering any context-specific factors. As a result, numerous similar agile adoption failures repeatedly happen.

In this research, we propose a **socio-intentional framework for agile methods tailoring** that focuses on the intentional (why) and social (who) dimensions. The framework targets precisely how team members work together to successfully adopt the agile practice and to achieve their goals. In this framework, we propose a methodology to analyze agile practices and to define the right strategies for adopting them based on the team's goals, its situations, and dependencies between team members. Every step in the process of analyzing an agile practice requires field information on the practice. To help practitioners get such information efficiently, we created a user-friendly tool which can provide the needed information. This tool was created on the basis of the agile practice adoption experiences found in the literature. With the information provided by the tool, we believe that the analysis should be done by means of graphical

models. The model provides a graphical illustration of the basic concepts and their relations that is easy to grasp even for non-experts on the modeling languages [142]. Models have been used to visualize, to communicate, and to better understand a system-to-be-built [24]. In our framework, we thus use a suitable modeling framework to ease the analyzing process. This modeling framework allows practitioners to visualize the relevant information about agile practice adoption reported in the literature in goal and social perspective. For instance, the goals we can achieve by adopting a practice, how team members should work collaboratively in a given situation for successful adoption, what problems are caused by team members, how to solve the problems, etc.

1.2 Research Design

In this section, we describe the research paradigm we follow to achieve the objectives of the thesis.

According to Hevner and March [73], two paradigms characterize much of the research in the Information Systems discipline: *behavioral science* and *design science*. While the goal of the former paradigm is the truth, the latter's is the utility. The *behavioral-science* paradigm has its roots in natural science research methods, which are used to develop and justify theories that explain or predict organizational and human phenomena surrounding the analysis, design, implementation, management, and use of information systems. Studies in behavioral science develop sets of concepts or specialized language with which we characterize phenomena. The results from these studies are used in higher-order constructions - such as laws, models, and theories - that make claims about the nature of reality [102]. On the other hand, *design science* attempts to create things that serve human purposes. *Design science* research cycle is fundamentally a problem-solving paradigm that is used to create and evaluate IT artifacts, which were created to solve identified organizational problems [102]. It seeks to create innovations that define the ideas, practices, technical capabilities, and products, through which the analysis, design, implementation, management, and use of information systems can be effectively and efficiently accomplished [47, 157].

As the objective of the thesis is to propose a framework to solve problems with the help of a tool, we thus follow the *design science* paradigm.

1.2.1 Design Science Research Paradigm

Hevner and March [73] provides *seven guidelines* as requirements for an effective design science research:

- *Problem Relevance:* design science research aims to acquire knowledge and understanding that enable the development and implementation of technology-based solutions to heretofore unsolved and important business problems;
- *Research Rigor:* design science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact;

- *Design as an Artifact:* design science research must produce a viable artifact. IT artifacts are broadly defined as constructs (vocabulary and symbols), models (abstractions and representations), methods (algorithms and practices), and instantiations (implemented and prototype systems);
- *Design Evaluation:* the utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods;
- *Research Contributions:* effective design science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies;
- *Design as a Search:* the search for an effective artifact requires utilizing available process means to reach desired ends, while satisfying laws in the problem environment;
- *Communication of research*: design science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

According to Hevner and Marchn [73], these seven guidelines are meant to assist the understanding of the requirements for effective design science research. They are however not mandatory and we must use our creative skills and judgment to determine when, where, and how to apply each criterion of the guidelines in a specific research project. Based on these guidelines, our work in this thesis can be summarized as in Table 1.1.

In the following sections, we describe what have been done answering the requirements of the design science.

1.2.1.1 Problem Relevance

The main purpose of the thesis is to address the lack of a socio-intentional framework that can efficiently help agile practitioners analyze agile practice and prepare for successful adoption. Within this work, three sub-problems are addressed:

1. **PR1 -** *Lack of verification about the relationship between the Agile Manifesto and agile practice selection*: in the topic of agile methods tailoring, many approaches have been proposed based on the Agile Manifesto (agile values and/or principles) [9, 18, 77, 83, 94, 100, 138]. Although the relationship between the Agile Manifesto and practice has been discussed in many researches, to the best of our knowledge, there is no formal verification on this matter yet. In addition, we have observed that many practitioners do not see the importance of the Agile Manifesto. In practice, development teams do not dedicate sufficient effort to understand agile values or principles before adopting agile methods [12, 16, 26, 48, 140]. This behavior coincides with one of the Agile Manifesto authors, Dave Thomas, [154], who claimed that "*Agile is Dead*". The lack of a formal verification leaves us questioning whether understanding the Agile Manifesto has any influence on choosing the right agile practice;

Table 1.1 Summary of research work based on seven guidelines for *Design science* research.

Seven Guidelines for Design Science	**Research Work**
Problem Relevance	- PR1: lack of verification about the relationship between the Agile Manifesto and agile practice selection; - PR2: lack of a system to recycle agile practice adoption experiences; - PR3: lack of a socio-intentional framework for agile methods tailoring.
Research Rigor	- RR1: conduct an SLR to study and verify the relationships between the Agile Manifesto and agile practices selection; - RR2: conduct another SLR to gather important knowledge related to agile practice adoption and then build an evidence-based ontology and a Graphic User Interface (GUI) tool; - RR3: propose a framework with a well-defined methodology to analyze agile practice and an illustrative example that allows justifying the framework applicability.
Design as an Artifact	- DA1: an ontology to describe entities, attributes, and relationships among knowledge concepts about agile practice adoption; - DA2: a GUI tool to help practitioners accessing the inserted knowledge with ease.
Design Evaluation	- DE1: validate ontology both theoretical (corpus-based) and empirical (survey) approaches; - DE2: evaluate the usability of the tool using survey;
Research Contributions	- RC1: a formal verification that Agile Manifesto is very important for practice selection; - RC2: a qualified evidence-based tool that can efficiently and effectively help agile practitioners understanding agile practices; - RC3: a socio-intentional framework that explains how to analyze agile practice and prepare for successful adoption, with the help of a tool and a modeling language.
Design as a Search	- DS1: our ontology contains enough knowledge to help practitioners understand the agile practices. It yet needs to be expanded using data from both the literature and more real-life case studies; - DS2: our tool is good enough in providing the information. It however needs to be improved and developed both usability and functionality; - DS3: our framework is proved to be applicable in real-life projects but we still need an empirical study to validate the framework.

Communication of research	- CR1: the verification of the relationship between the Agile Manifesto and agile practices selection can encourage practitioners to have a deep understanding of the Agile Manifesto before their adoption. It also provides a clear-cut validation on the relationship which can be used in future research, rather than an assumption; - CR2: the ontology and the tool allow practitioners to quickly understand agile practice adoption. They can also help the researchers know how to create an ontology to recycle and to share knowledge; - CR3: the framework allows practitioners to gain knowledge as a big picture and plan for successful adoption. It provides a clearer understating of the socio-intentional aspect of agile methods tailoring for both practitioners and researchers.

2. **PR2 -** *Lack of a system to recycle agile practice adoption experiences*: over the years, countless experiences about the agile methods or practice adoptions have been shared in the literature. These experiences should be helpful for practitioners in understanding agile practices on a deeper level and allow them to prepare for the successful adoption. As this knowledge is unstructured, it thus requires lots of effort and time to locate relevant information. To make the knowledge more accessible, Esfahani & Yu [58] gathered information about (1) the goals that can be achieved by a practice and (2) the requisites required for its adoption, and store in a repository. Even though the repository can help practitioners understand two concerns related to agile practices (goal and requisite), many remaining concerns need to be recycled in a systematic manner;
3. **PR3 -** *Lack of a socio-intentional framework for agile methods tailoring*: Even though humans and their interaction are known as one of the foundations in agile methods, there are not many researches that focus on the social aspect within their tailoring process. To the best of our knowledge, there are only two researches that advocate the use of socio-intentional modeling techniques to depict social aspects of agile methods [56, 59]. The former research [56] shows the advantage of modeling social aspects, it however does not provide a clear methodology of how to build the models and how to use these models effectively to analyze agile practice. The latter research [59] proposed a methodology for tailoring agile methods, it however only focuses only on the *goal* aspect.

1.2.1.2 Research Rigor

Our research objective can be achieved by solving the above-mentioned problems, using well-defined research protocols.

1. **RR1 -** To study and verify the relationships between the Agile Manifesto (value and principle) and agile practices selection, we conducted an SLR

by following the approach described by Kitchenham and Charters [82]. We extracted problems, expectations, and the benefits which are the reasons behind agile methods or practice selection from the selected articles. We then compared these problems, expectations, and benefits with the agile values and principles. The results from the comparisons allow us to formally confirm the relationship between the Agile Manifesto and agile practice selection;

2. **RR2 -** To recycle the knowledge in a systematic manner, we started by conducting another SLR by following the same approach [82]. We exhaustively gathered all the necessary information related to the agile practice adoption, based on experiences reported in research papers. Using the collected knowledge, we then created an ontology by basically following the methodology proposed in [109]. Finally, we validated the ontology to ensure that all concepts and relationships correctly represent the purpose for which it was created. We opted for two validation techniques: (1) corpus-based approach [21] and (2) a survey with agile experts;

3. **RR4 -** To propose a new socio-intentional framework for agile methods tailoring, we defined a methodology that focuses on both the *why* and *who* dimensions. With our well-defined set of steps, practitioners can define the right strategies to achieve their goal, know how to coordinate the activities of the various actors and how they depend on each other. To be efficient, we also included how to get the knowledge from our tool, how to visualize them in models and how to analyze the result in every step of the methodology. To prove that our framework is applicable in real-life projects, we applied it to a real case as an illustrative example.

1.2.1.3 Design as an Artifact

We built two artifacts to solve the addressed problems.

1. **DA1 -** An efficient solution to recycle the experiences of agile practices adoption would be an approach that allows us to describe the knowledge in a way that can be systematically reusable. In this thesis, *ontology* is the key to our solution. The notion of ontology refers to a consensus that defines the entity, attribute, and relationship among knowledge concepts within a specific domain using explicit descriptions and specifications in an interoperable format and understandable by both humans and machines [35, 52]. Ontologies have been widely used to represent empirical knowledge in a structured manner. They allow sharing, reusing, and supporting decision making [36, 124, 34]. The notion and advantages of ontology make it a prominent solution to recycle agile practice adoption knowledge;

2. **DA2 -** Using an available tool to retrieve data placed within the ontology requires some preliminary knowledge. For user convenience, we thus created a GUI tool using Python programming language, named "OBAMA - Ontology-Based tool for Agile Methods Adoption". Our tool is in a notebook-style where each page serves for a functionality. It can answer

15 concerns related to agile practice adoption. Users can choose to see all the information related to the agile practice adoption that we have inserted, and also filter for the relevant information by describing their adoption goals, and team's situations.

1.2.1.4 Design Evaluation

There is no way to evaluate the result of an SLR, but we ensure the quality of the results by addressing the threat to the validity as much as possible. The remaining parts that need to be evaluated are ontology, tool, and the framework.

1. **DE1 -** Among many approaches to evaluate an ontology [27, 28, 122, 125], we chose to follow Rao and Lila's assessment part of their framework [125]. Their assessment part is the most suitable for our case due to its similarity to the methodology we used for our ontology creation. Inspired by their assessment part, three aspects were formulated to validate our ontology (1) Are the concepts and/or relationships in the ontology used to answer all concerns and vice versa? (2) Are all the concerns relevant? and (3) Are the answers to the concerns correct? As our ontology is created based on a text corpus of the domain, i.e., scientific research papers on agile practice adoption, we validate the first aspect using the corpus-based approach. For the other two aspects, we validate using a survey to directly get the answers from agile experts;

2. **DE2 -** Using the same survey, we also evaluated our tool with nine questions related to its usability in providing the information and helping practitioners finding the right agile practices for adoption.

1.2.1.5 Research Contributions

In this thesis, three main contributions were made to solve the targeted problem:

1. **RC1 -** The results from our first SLR allow us to understand how the Agile Manifesto has been discussed in tailored agile methods adoption and to verify whether agile practices selection can be related to agile values or principles defined in the Agile Manifesto. The result shows that *development teams have lost attention on the Agile Manifesto*, having less than half of them mentioned the Agile Manifesto. On the contrary, by comparing the 4 values and 12 principles of the Agile Manifesto with the team's problems, expectations, and benefits extracted from the literature, at least 80% of them can be mapped to each other. This result allows us to verify that *the Agile Manifesto is highly relevant to the reasons behind the agile adoption.* As it still covers fundamental aspects of any agile methodology, it is important for the team to understand the Agile Manifesto before adopting agile methods;

2. **RC2 -** We have built a qualified evidence-based tool that can efficiently and effectively help agile practitioners understanding agile practices. By conducting another SLR, we exhaustively extracted 86 case studies on

agile practice adoption. Using these case studies, we created an ontology to support knowledge representation about agile practice adoption. In addition, we added seventeen inference rules to systematically discover more relationships among concepts in the ontology. After that, we theoretically validated our ontology by following the corpus-based approach [21]. The result shows that our *ontology represents accurately the information related to the agile practice adoption with minimum refinement for any unseen knowledge in the future*. We then created a user-friendly tool using Python programming language before we validated both ontology and tool using a survey. The results from our survey show that *our ontology and tool can provide the information efficiently, effectively and it helps a team decide if they should adopt a practice*. Even though our tool cannot fully satisfy experts, yet they agree that it is good enough to serve our purpose;

3. **RC3 -** We have defined a clear methodology for agile methods tailoring that consists of two levels. *Tactical level* allows practitioners to (1) identify the best practices to adopt based on their goals and (2) check the suitability of a team based on their situations. *Operational level* allows the team to identify (1) the vulnerabilities during the practice adoption caused by team members, (2) possible problems based on the experiences, and (3) solutions to avoid the vulnerabilities and solve the problems. In every step of this framework, we explain how practitioners can use our tool to get the information and then visualize them in models to facilitate the analyzing processes.

1.2.1.6 Design as a Search

While our defined objectives have been achieved with both theoretical and empirical validations, some improvements are still possible.

1. **DS1 -** Our ontology model and knowledge always need to be expanded using data from both the literature and real-life case studies. In the first SLR that we conducted, we collected data related to only the five most commonly used practices. This amount is still very small compared to the number of existing agile practices. Apart from the literature, the ontology does not include any case study in real life. We believe that there is valuable knowledge that can only be extracted by observation;

2. **DS2 -** There are parts of the tool that need to be improved and developed in both usability and functionality. For instance, in the current version of our tool, users can get the information only by selecting the concerns they are interested in. A more sophisticated tool would allow users to switch easily from one concept or concern to another. Also, when users want to learn more about the source of the information, they should be able to reach them within a simple click. More than that, a complete tool would allow the users to encode new knowledge easily and to evaluate the reliability of the existing knowledge;

3. **DS3 -** Our framework is dedicated to only the planning part by focusing on how to prepare a team before agile practice adoption, while there should also include the evaluation process to further check the adoption success. Our framework also lacks a study with the real team that allows validating how helpful this framework is for the team in adopting agile practices. We need empirical validation for instance by an exploratory study to validate our framework.

1.2.1.7 Communication of Research

The contributions of our research are beneficial for both practitioners and researchers.

1. **CR1 -** The verification of the relationship between the Agile Manifesto and agile practices selection encourages practitioners to have a deeper understanding of the Agile Manifesto before their adoption. It explains why the Agile Manifesto is important by showing how it can help maximize the team's expectations and, eventually the benefits of agile adoption. For the researcher, these results provide a clear-cut validation on the relation between the Agile Manifesto and agile practices, instead of being just an assumption or belief. This validation can be used as evidence to propose any approach or framework that can help improve agile practice adoption;

2. **CR2 -** The ontology and tool allow practitioners to quickly learn about useful concepts related to agile practices and their relationships. The information provided by the tool can help (1) the team to find the right practices suitable for the team and (2) understand how to avoid the risk during adoption. Researchers can use it to discover or/and understand different aspects of agile practice adoption. They can also learn how to use ontology as an effective solution to recycle and to share knowledge among practitioners in a structured and exploitable way;

3. **CR3 -** This framework emphasizes an important perspective in the agile methods tailoring by combing the social with the goal aspect together. By following our framework, the practitioners can understand how to collaboratively work together and also the motivations and rationale behind their activities. At the same time, the practitioners can make good use of the agile practice adoption experiences which can be found in the literature. This framework also shows the researchers the advantages of using the diagram to visualize the information to ease the analyzing process.

1.3 Reading Roadmap

This thesis consists of four parts and nine chapters. Figure 1.1 provides the structure and reading roadmap of the thesis. Under the title of each chapter, we list what we addressed in that chapter based on the seven guidelines of design science, followed by the name of the conference in which the content of the chapter is published or under reviewed.

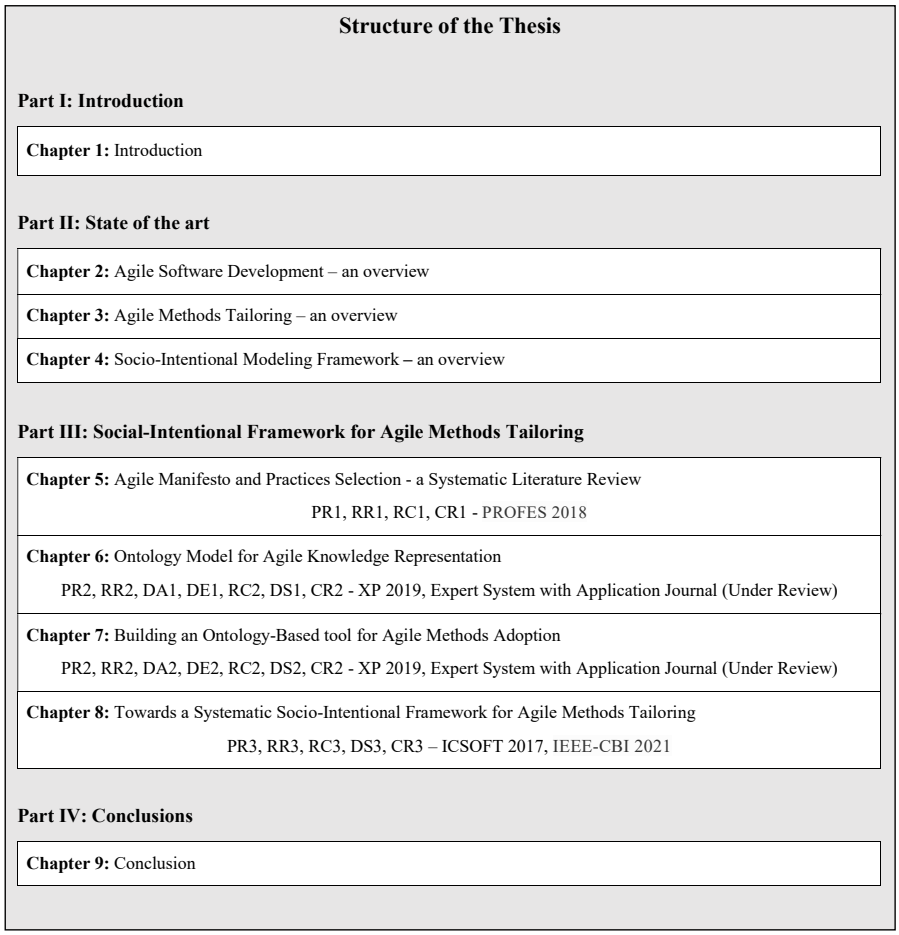


Fig. 1.1 Thesis reading roadmap.

The second part of this thesis provides a general literature review; Chapter 2 provides the background of agile software development methodologies and an overview of the current most commonly used agile methodologies, Chapter 3 provides a basic review of different approaches for software methods tailoring in general and particularly for agile methods. In Chapter 4, we review some modeling frameworks related to either goal or social dimension. The purpose of each chapter in this first part is to develop a general overview and understanding of the subject without relating each element of theory to a specific scientific contribution of the thesis.

The third part of this thesis is built as a set of individual contributions presented individually from Chapter 5 to Chapter 8. In each chapter, the specific context, related work, contributions, research protocol, and the result are discussed within the chapter themselves.

Chapter 5 aims at understanding how has the Agile Manifesto and its importance been discussed in tailored agile methods adoption. It also aims at verifying whether the Agile Manifesto and agile practices selection are related. In this chapter, we explain how we conducted the SLR, including details on research questions, search strategy, and data extraction. After, we present the results of our literature review, followed by the threads to validity. Finally, we summarize a conclusion and findings.

Chapter 6 aims at building a valid ontology that allows to systematically recycle the knowledge about agile practices adoption found in the literature. In this chapter, we describe how we conducted another SLR to gather knowledge from the literature. We then explain how we built and theoretically validated the ontology. Finally, we describe how we conducted the survey to empirically validate the ontology with the results.

Chapter 7 aims at building a user-friendly tool that allows practitioners to efficiently retrieve the knowledge from the ontology. We start by explaining the importance of a user-friendly tool and giving general information about the techniques needed for the development. We then explain in detail each important component needed for developing the tool. Finally, we explain how we evaluated our tool with agile experts using a survey.

Chapter 8 aims at proposing a socio-intentional framework for agile methods tailoring with the help of our tool and modeling techniques. We start by explaining how to find the right modeling technique to represent the knowledge of agile methods adoption. We then describe the methodology for tailoring the agile methods. Finally, we use a case of a real software development team as an illustrative example to demonstrate how our framework is applied in real life.

Chapter 9 concludes the thesis by discussing the contributions, limitations, and the future works.

Part II

State of the Art

Chapter 2

Agile Software Development

In the early software development era, software was developed sequentially by following the methodology called the *waterfall model*. This methodology was described as a set of phases where each of them can be started only when the previous one is completely done and the delivering phase is put at the very end of the development. This kind of process is too rigid and too risky for today's user-intensive software development. The *Agile* movement has emerged to effectively deal with quality expectations, to ensure that the software is solving the right problems, and to be able to deal with immediate changes in the requirements.

This chapter provides an introduction to agile software development, its foundation, and some popular agile methodologies. This chapter is structured as follows. Section 2.1 exposes the foundation of agile methodologies called the Agile Manifesto. Section 2.2 describes some of the most popular agile methodologies based on the most recent survey, such as Scrum, eXtreme Programming (XP), Kanban, etc. Finally, we conclude the chapter in Section 2.3.

2.1 Foundation of Agile Methodologies

Since the emergence of software in the 1950s, various software development methodologies have been proposed. They aim to reduce cost and to meet users' needs with an adequate level of quality. Among the methodologies which were proposed at the earlier time, also known as *traditional methodologies*, Waterfall [128], V-model [64], Spiral model [22], Rapid Application Development (RAD) [103] and Rational Unified Process (RUP) [88] can be considered as the popular ones. These methodologies are characterized as plan-driven, document-oriented, process-oriented, and based on formal communication [4, 92, 119, 143]. In other words, in these methodologies, the software is built using a pre-defined and existing plan. These methodologies work well when the requirements are complete and stable. However, they do not respond well to user-intensive software where the requirements are likely to evolve.

To avoid the problems of the traditional methodologies, practitioners started mixing old and new ideas to create new methodologies that worked in a given situation of their teams. These methodologies emphasized close collaboration between the Development Team and business stakeholders; frequent delivery of

business value, tight, self-organizing teams; and smart ways to craft, confirm, and deliver code. These new methodologies were known as *Agile* and they were created before 2001. It includes eXtreme Programming (XP) [17], Scrum [136], Dynamic Systems Development Method (DSDM) [144], Adaptive Software Development (ASD) [74], Crystal [39], Feature-Driven Development (FDD) [114] and Pragmatic Programming [153].

To find the consistency and give a canvas to these agile methodologies, 17 representatives of these methodologies met at Snowbird in Utah in 2001. The purpose of this meeting was to discuss and establish a common ground for an alternative to the structured and traditional heavy software development life cycles. What emerged from the meeting was a manifesto for Agile Software Development, commonly known as the *Agile Manifesto*[1] and the creation of the *Agile Alliance* as a guiding force for agile practitioners.

In this manifesto, *four values* and *twelves principles* were defined, making Agile a value and principle based rather than rule-based movement [66, 63].

2.1.1 Agile Values

The Agile Manifesto [66] states that, with an agile mindset, items on the left ought to be valued more than items on the right, i.e., Individuals and interactions (left item) over processes and tools (right item). However, those values are the preferences, not the alternatives, and we should not undervalue the latter either. Exploring each of these values will help in gaining knowledge of the agile process philosophy while exposing how applying the philosophy to define methodologies will enhance software development, aligning it with today's volatile markets. These **four** agile values include:

1. **Individuals and interactions** over processes and tools: individuals are more flexible and responsive to changes while processes are rigid and scheduled. The key to a good workflow and a high-quality product are teamwork, communication, and collaboration;

2. **Working software** over comprehensive documentation: one of the big differences between Agile and Heavyweight is the amount of the document produced during the development [150]. Documentation should only be needed in very few and specific instances, and if a software works well, there is no need for documentation;

3. **Customer collaboration** over contract negotiation: it is fundamental in agile that methodologies to have a close collaboration between the team and their customer to understand exactly what they need. Negotiating a contract makes the process inflexible;

4. **Responding to change** over following a plan: the benefits of having a plan are not questioned, but more importantly the ability to adapt to the rapidly changing environment and requirement.

[1] http://agilemanifesto.org/

2.1.2 Agile Principles

The Agile Manifesto [66] documented 12 principles to guide the team and manager rather than following pre-defined rules. Diebold and Zehler [49] defined agile principles as the high-level ideas behind agile software development as refinements of the core values defined in the Agile Manifesto. The role of the principles was described as a bridge that narrows the gap between the abstract general values and detailed specific practices [79, 99]. These **twelves** agile principles include:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software;
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage;
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale;
4. Business people and developers must work together daily throughout the project;
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.;
6. The most efficient and effective methodology for conveying information to and within a Development Team is a face-to-face conversation;
7. Working software is the primary measure of progress;
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely;
9. Continuous attention to technical excellence and good design enhances agility;
10. Simplicity, the art of maximizing the amount of work not done is essential;
11. The best architectures, requirements, and designs emerge from self-organizing teams;
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

2.2 Overview of the main Agile Methodologies

The popularity of agile methodologies is constantly increasing to offer flexible software development. Over the last two decades, many agile methodologies have been proposed with their set of values, principles, and practices to meet specific requirements and contexts. For instance, Scrum is proposed to put more focus on project management organization, while XP is designed to be more responsive to customer requirement changes [105].

According to the 13th survey report conducted by the VersionOne in 2019 [161], the most commonly used agile methodologies include: Scrum, Extreme Programming (XP), Scrumban, Kanban, Lean, and the combinations of these methodologies (see Figure 2.1). In the following sections, we describe these methodologies individually. We also show the relationships between the Agile Manifesto and each agile methodology. This relationship is the result of mapping between the values and principles of each methodology with those defined in the Agile Manifesto.

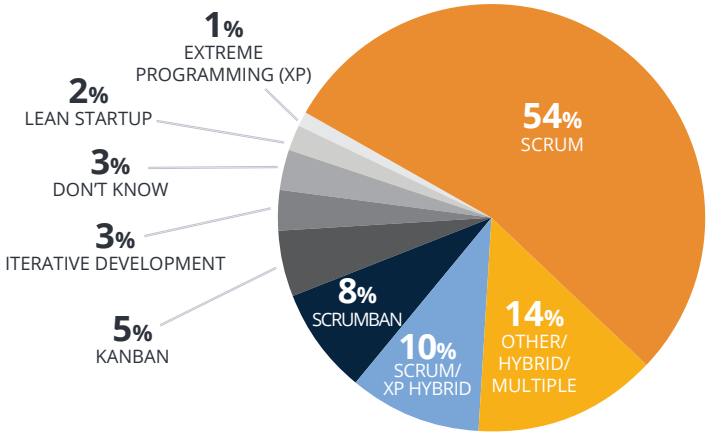


Fig. 2.1 Most used agile methodologies based on VersionOne's 13th survey (from [161])

2.2.1 Scrum

Scrum was first introduced by Schwaber in an article in 1996 [134]. But only until 2002 that the methodology was fully described in a book written by Schwaber and Beedle [136]. Scrum is a process framework to deliver products with the highest possible value and handle complex problems or situations. Bogojevi'c [23] defined Scrum as a lightweight software development process having a cross-functional team who develop as much quality software as possible within a series of short time boxes called "Sprints", which last about a month. Scrum is characterized by short, intensive, daily meetings of software development stakeholders. Compared to other agile methodologies, Scrum focuses on project management rather than technical practices.

Scrum was originally created with only practices. Later on, some authors defined its values and principles based on experiences. According to Sutherland and Schwaber [151], Scrum is founded on empirical process control theory or empiricism, where the knowledge comes from experience and making decisions based on what is known. The empiricism theory is supported by **three** pillars: *transparency, inspection,* and *adaptation.*

2.2.1.1 Scrum values

The **three** pillars that support empiricism can come to life when **five** Scrum values are embodied and lived by the Scrum team. These **five** values include: *commitment, courage, focus, openness and respect.* Agile Alliance [8] explains each value as follows:

- *Commitment*: team members personally commit to achieving team goals;
- *Courage*: team members do the right thing and work on tough problems;
- *Focus*: concentrate on the work identified for the sprint and goals of the team;
- *Openness*: team members and stakeholders are open about the works and challenges that the team encounters;
- *Respect*: team members respect each other to be capable and independent.

2.2.1.2 Scrum Principles

Agile Alliance [8] also provides the agile team with **three** principles underpinning the empirical nature of Scrum with the brief description as follows:

- *Transparency*: the team must work in an environment where everyone is aware of what issues other team members are running into. Teams surface issues within the organization, often ones that have been there for a long time that get in the way of the team's success;
- *Inspection*: frequent inspection points were put into the framework to allow the team an opportunity to reflect on how the process is working. These inspection points include the Daily Scrum meeting and the Sprint Review Meeting;
- *Adaptation*: the team constantly investigates how things are going and revises those items that do not seem to make sense.

Another group called Scrum Study [137] also defines extra **six** principles as follows:

- *Empirical Process Control*: this principle emphasizes the core philosophy of Scrum based on the three main ideas of transparency, inspection, and adaptation;
- *Self-organization*: this principle focuses on today's workers who deliver significantly greater value when self-organized and this results in better team buy-in and shared ownership, and an innovative and creative environment which is more conducive to growth;
- *Collaboration*: this principle focuses on the three core dimensions related to collaborative work: awareness, articulation, and appropriation. It also advocates project management as a shared value-creation process with teams working and interacting together to deliver the greatest value;

Table 2.1 Mapping Scrum values and principles with the Agile Manifesto

Agile Manifesto	Scrum
Values	
- Individuals and interactions over processes and tools	Commitment, Courage, Respect
- Working software over comprehensive documentation	Focus
- Customer collaboration over contract negotiation	Openness
Principles	
- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software	Value-based prioritization
- Deliver working software frequently	Time-boxing, Iterative Development
- Business people and developers must work together daily throughout the project	Transparency, Collaboration
- Build projects around motivated individuals	Self-organization
- The most efficient and effective methodology of conveying information to and within a Development Team is face-to-face conversation	Transparency
- Continuous attention to technical excellence and good design enhances agility	Adaptation
- At regular intervals, the team reflects on how to become more effective	Inspection

- *Value Based Prioritization*: this principle highlights the focus of Scrum to deliver maximum business value from beginning early in the project and continuing throughout;
- *Time-boxing*: this principle describes how time is considered a limiting constraint in Scrum and used to help effectively manage project planning and execution. Time-boxed elements in Scrum include Sprints, Daily Stand-up Meetings, Sprint Planning Meetings, and Sprint Review Meetings;
- *Iterative Development*: this principle defines iterative development and emphasizes how to better manage changes and build products that satisfy customer needs. It also delineates the Product Owner's and organization's responsibilities related to iterative development.

Table 2.1 list the results from mapping Scrum values and principle with the Agile Manifesto.

2.2.1.3 Scrum Roles

Following Schwaber et al. [151], Scrum Teams are self-organizing and cross-functional which consists of a Product Owner, the Development Team, and a Scrum Master.

- *Product Owner*: it is not a committee but only one person who is responsible for maximizing the value of the product and managing the Product Backlog. Product Owner may represent the desires of a committee in the Product Backlog, but those who want to change a Product Backlog item's priority must address it to the Product Owner;
- *Development Team*: it is a group of 4 to 10 cross-functional people who are responsible for delivering a potentially releasable product at the end of each sprint. In Scrum, the Development Team is structured and empowered by the organization to organize and manage their own work;
- *Scrum Master*: it is the person who is an expert at Scrum and who can therefore coach others. The Scrum master is the role who is responsible for ensuring the team lives agile values and principles and follows the processes and practices that the team agreed that they would use.

2.2.1.4 Scrum Artifacts

Scrum's artifacts represent work or value to provide transparency and opportunities for inspection and adaptation. Artifacts allow maximizing transparency of key information to make everyone have the same understanding of the artifact. Following [8, 151], there are four artifacts in Scrum:

- *Product Backlog*: it is an ordered list of all the possible changes that could be made to the product. Items on the product backlog are options, not commitments. Just because an item exists in the Product Backlog, it does not guarantee it will be delivered. The Product Owner maintains the product backlog on an ongoing basis, including its content, availability, and ordering;
- *Sprint Backlog*: it is a set of selected product backlog items that a team must deliver by the end of each sprint. The team needs to identify necessary tasks to develop those sprint backlog items in order to achieve the sprint goal;
- *Increment*: it is a collection of the product backlog items that meet the team's "Definition of Done" by the end of the sprint. The Product Owner may decide to release the increment or build upon it in future sprints;
- *Definition of Done*: it is a team's shared agreement on the criteria that a product backlog item must meet before it is considered done.

2.2.1.5 Scrum Practices

The event in Scrum is a formal opportunity to inspect and adapt something which is specifically designed to enable critical transparency and inspection [151]. There are 4 events in Scrum which can be performed within each iteration, also known as *Sprint*. These four events include:

- *Sprint Planning*: it is an event where the entire Scrum team collaborates to plan the work to be performed within the next Sprint. Sprint Planning is time-boxed to a maximum of eight hours for a one-month Sprint. The result from the Sprint planning is the list of what can be delivered from the upcoming Sprint and how these works should be done;
- *Daily Meeting*: it is a 15-minute time-boxed event for the Development Team which is held every day to plan work for the next 24 hours. Three main questions that should be asked during the events include "what did I do yesterday that helped the Development Team meet the sprint goal?", "what will I do today to help the Development Team meet the sprint goal?", and "do I see any impediment that prevents me or the Development Team from meeting the sprint goal?";
- *Sprint Review*: it is held at the end of the sprint where the Scrum Team and stakeholders collaborate to inspect what was done in the sprint and adapt the Product Backlog if needed. The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next sprint;
- *Sprint Retrospective*: it is conducted after the Sprint Review and prior to the next Sprint Planning. It is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next sprint. The purpose of the event is to inspect how the last sprint went, to identify and order the major items that went well and potential improvements, and to create a plan for implementing improvements in the next sprint.

2.2.2 eXtreme Programming (XP)

XP is an agile methodology founded by Kent Beck based on an experiment of the Chrysler C3 Project in 1996 [23]. According to Beck [17], XP is a lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop software. XP is suitable for the Development Team with the size of 2 to 12 members and preferably co-located. This methodology focuses on the technical aspect consisting of 10 to 12 engineering practices such as unit tests, frequent full system integration, pair programming, simple design, and frequent releases of working software, etc. Each iteration that usually lasts between 1 and 3 weeks consists of 6 phases including: exploration, iteration planning, iteration to release, production, maintenance, and death phase.

2.2.2.1 XP Values

According to Beck [17], **four** values of XP were defined as follows:

- *Communication*: everyone is part of the team and face-to-face is the best way to transfer knowledge between each other in the team;
- *Simplicity*: we need to avoid waste by doing only what is needed and asked for, do not try to predict the future;

- *Feedback*: concrete feedback about the current state of the system is absolutely priceless. The more feedback we have, the easier it is to communicate;
- *Courage*: courage is an effective action in the face of fear. A team needs the courage to raise organizational issues, to stop doing something that does not work, and to accept the feedback.

2.2.2.2 XP Principles

Following Beck [17], there are **five** fundamental principles as follows:

- *Rapid feedback*: team members should be able to get the feedback as soon as possible. They then need to understand and react to it right away, within seconds or minutes instead of days, weeks, or months;
- *Assume simplicity*: treat every problem as if it can be solved with ridiculous simplicity. Team should focus on doing a good job (tests, refactoring, communication) of solving today's job today and trust their ability to add complexity in the future where they need;
- *Incremental change*: big changes that are made all at once just do not work. Any problem is solved with a series of the smallest changes that make a difference;
- *Embracing change*: the best strategy is the one that preserves the most options while actually solving the most pressing problem;
- *Quality work*: everybody likes doing a good job. In order to enjoy their works, the team needs to do an excellent job and make valuable products.

Table 2.2 list the results from Mapping XP values and principle with the Agile Manifesto.

2.2.2.3 XP Roles

Kent [17] describes seven roles in XP as follows:

- *Programmers*: this role is responsible for programming the entire system, writing tests, and keeping the program code as simple as possible;
- *Customer*: this role is responsible for writing the stories and functional tests, prioritizing the stories, giving feedback on the result, and deciding when each requirement is satisfied;
- *Tester*: this role is responsible for writing test cases, running functional tests, broadcasting test results, and maintaining testing tools;
- *Tracker*: this role is responsible for giving feedback about everyday work on a project. The Tracker traces time estimates made by the team and provides feedback on how accurate they are, inspect the progress of each iteration, and evaluate whether the iteration goal is reachable;

Table 2.2 Mapping XP values and principle with the Agile Manifesto

Agile Manifesto	XP
Values	
- Individuals and interactions over processes and tools	Communication, Courage
- Working software over comprehensive documentation	Simplicity
- Customer collaboration over contract negotiation	Feedback
Principles	
- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software	Incremental change
- Welcome changing requirements	Embracing change
- Business people and developers must work together daily throughout the project	Rapid feedback
- Continuous attention to technical excellence and good design enhances agility	Quality work
- Simplicity, the art of maximizing the amount of work not done is essential	Assume simplicity

- *Coach*: it is someone who has a sound understanding of XP practices and experience in working with XP teams. Coach is responsible for guiding the other team members in following the process;
- *Consultant*: it is an external member of the team. His primary role is to advise the team with his technical knowledge and to guide the XP team in solving their specific technical issues;
- *Manager*: this role is responsible for making decisions regarding the project, communicating with the project team in order to understand the status of the project, and distinguishing any difficulties or deficiencies in the process.

2.2.2.4 XP Artifacts

Following Beck [17], there are five artifacts produced within an XP project as follows:

- *User Story Card*: it is an index card that contains a requirement in the form of a user story, together with a short description of requirements, priority, effort, and test scenarios. The user story card is used for implementation, discussion, and planning;
- *Task List*: it is a listing task needed to be built in order to accomplish a user story. This task list helps programmers to estimate the effort and planning;

- *CRC Card*: it stands for Class-Responsibility-Collaboration. It contains the responsibilities and collaborations of classes (Object-Oriented) that allow the team to do the design of the system;
- *Customer Acceptance Test*: it is the test scenario that customers write to validate the implementation. Normally, it is written on the user story card;
- *Visible Wall Graphs*: it is a graphical board that is used to publicly show the progress of the team — e.g., how many user stories are being developed, tested, and accepted. It allows the team to communicate and see the progress of the project.

2.2.2.5 XP Practices

Below are the descriptions of the practices as described by Kent [17]:

- *Sit Together*: team members are asked to sit together in the same space allows them to have face-to-face communication without barriers;
- *Whole Team*: build a cross-functional group of people with the necessary roles to work together daily to accomplish a specific outcome;
- *Informative Workspace*: set up a team space to facilitate face-to-face communication, allow people to have some privacy when they need it, and make the work of the team transparent to each other and interested parties outside the team;
- *Energized Work*: do not overwork yourself or let others overwork you, take steps to make sure you are able physically and mentally to get into a focused state;
- *Pair Programming*: all software products are developed by two people sitting at the same machine. The idea behind this practice is that two brains and four eyes are better than one brain and two eyes. You effectively get a continuous code review and quicker response to nagging problems that may stop one person dead in their tracks;
- *Stories*: it is a short description of things users want to be able to do with the product. Stories are used for planning and served as reminders for more detailed conversations when the team gets around to realizing that particular story;
- *Weekly Cycle*: it is synonymous with an iteration. The intent behind the time-boxed delivery period is to produce something to show to the customer for feedback. In the case of XP, the team meets on the first day of the week to reflect on progress to date, the customer picks the stories they would like delivered in that week, and the team determines how they will approach those stories;

- *Quarterly Cycle*: it is synonymous with a release. The purpose is to keep the detailed work of each weekly cycle in the context of the overall project. The customer lays out the overall plan for the team in terms of features desired within a particular quarter, which provides the team with a view of the forest while they are in the trees, and it also helps the customer work with other stakeholders who may need some idea of when features will be available;
- *Slack*: the idea behind slack in XP terms is to add some low priority tasks or stories in the weekly and quarterly cycles that can be dropped if the team gets behind on more important tasks or stories. In other words, account for the inherent variability in estimates to make sure that the team have a good chance of meeting the forecasts;
- *Ten-Minute Build*: the goal with the Ten-Minute Build is to automatically build the whole system and run all of the tests in ten minutes. This practice encourages the team to automate their build process so that they are more likely to do it on a regular basis and to use that automated build process to run all of their tests;
- *Continuous Integration*: it is a practice where code changes are immediately tested when they are added to a larger code base. The benefit of this practice is they can catch and fix integration issues sooner;
- *Test-First Programming*: instead of running the test after developing code and write the test, in this practice developer should follow the path: write failing automated test $\rightarrow$ run failing test $\rightarrow$ develop code to make the test pass $\rightarrow$ run test $\rightarrow$ repeat;
- *Incremental Design*: do a bit of work upfront to understand the proper breadth-wise perspective of the system design, and then dive into the details of a particular aspect of that design when they deliver specific features. This approach reduces the cost of changes and allows the team to make design decisions when necessary based on the most current information available.

2.2.3 Kanban

According to Kniberg et al. [87], Kanban is taken from a Japanese term which means *Signboard*. A Kanban system utilizes visual cues that calculate what to produce when to produce and how much to produce. Kanban is intended to manage the workflow and increase performance which aims to focus on the efforts on the items that bring value to the end customer and remove waste while not overburdening the Development Team. Like Scrum, Kanban is a process designed to help teams work together more effectively by using concepts such as *wide communication, signboard* and *working status* to provide a comprehensive view of the project. Opposite to Scrum, instead of working in a time-box of work every 2 to 4 weeks, Kanban encourages continuity in the workflow. There is not any defined role in Kanban, practitioners can define as many roles as they want. However, for each iteration, Kanban recommends minimizing the cycle

time, so if adding a role helps minimize the cycle time, the role can be added and if it makes the process slower, then the role should not be there. Also, if the cost of the role is higher than the value of improved cycle time, then it is an unnecessary role.

2.2.3.1 Kanban Values

Mike Burrows [30] defines **nine** values of Kanban as follows:

- *Understanding*: understand the process that had been working and what they are about to change;
- *Agreement*: agree to move forward and accept the changes;
- *Respect*: respect each other's roles and responsibilities;
- *Leadership*: encourage and support leadership and initiative at all levels within an organization by facilitating self-organization;
- *Flow*: make ongoing process with a constant pace by understanding the effort to get certain results in specific ways;
- *Customer Focus*: the actual value of a project is the completed tasked which is the utmost satisfaction of the customer;
- *Transparency*: make teams transparent with 3 principles: visualization of process, the introduction of explicit policies, and creating feedback loops;
- *Balance*: maintain the work balance to keep both the employees and the client happy by limiting the number of work in progress (WIP);
- *Collaboration*: work together and search beyond our inner team to collaboratively find solutions and make plans for organizational and process improvements.

2.2.3.2 Kanban Principles

According to Agile Alliance [7], **six** principles in Kanban are defined as follows:

- *Start with what you do now*: understand current processes as they are actually practiced and respect existing roles, responsibilities, and job titles;
- *Agree to pursue improvement through evolutionary change*: accept and welcome changes in a continuous learning process;
- *Encourage acts of leadership at every level*: empower the team members and give room for leadership at each level of the organization;
- *Understand and focus on your customers' needs and expectations*: prioritize on activities that fulfill the client's needs and expectations and then create true business value;

Table 2.3 Mapping Kanban values and principle with the Agile Manifesto

Agile Manifesto	Kanban
Values	
- Individuals and interactions over processes and tools	Respect, Leadership, Flow, Balance
- Customer collaboration over contract negotiation	Customer focus, Transparency, Collaboration
- Responding to change over following a plan	Agreement
Principles	
- Business people and developers must work together daily throughout the project	Understand and focus on your customers' needs and expectations
- Build projects around motivated individuals	Encourage acts of leadership at every level
- The best architectures, requirements, and designs emerge from self-organizing teams	Manage the work; let people self-organize around it
- At regular intervals, the team reflects on how to become more effective	Agree to pursue improvement through evolutionary change, Evolve policies to improve customer and business outcomes

- *Manage the work; let people self-organize around it*: let people manage their way to achieve it, they know best how to do their work;
- *Evolve policies to improve customer and business outcomes*: as the environment evolves, rules and policies should evolve with it.

Table 2.3 lists the results from Mapping Kanban values and principles with the Agile Manifesto.

2.2.3.3 Kanban Practices

Based on Baleviciute [15], there are five practices in Kanban:

- *Visualize Workflow*: create an overview of the entire project that encourages a team to seek the best result using the board. Divide the board into three sections: "Input", "Work In Progress" and "Output". Put task cards on the board and tag important tasks with short descriptions. Use different columns to show team members which tasks are the most important and should be started first;
- *Create a Workflow*: backlog can be split into two columns: backlog and backlog priorities. Assign stories or tasks to team members by the "Pull principle" where every team member chooses his own task. Tasks that are in the priority column are pulled first. The column "Work In Progress"

can have other columns such as “Plan”, “Development”, “Design”, “Draft”, “Test”, “Deploy”, “Integrate”, “Done”,.etc;

- *Work In Progress (WIP) Limits*: there should always be work limits per column to avoid multi-tasking which may result in a wasted time. WIP limits helps to match the team's development capacity. If the given column is set to four, then there should be only four tasks being worked on at one time. Those tasks that cannot be completed should be moved back to the backlog and a new task is selected from the priority list. WIP limits is great to identify bottlenecks;
- *Manage Performance*: Kanban uses lead and cycle times to measure performance. Lead time shows how long it takes to complete a task from its request until that request is done and delivered to the end consumer. Cycle time shows how long a task has been in production or the work in progress section except for the queue. A cumulative flow diagram helps to monitor the progress of all tasks;
- *Planning Routines*: Kanban has neither a precise planning routine nor pre-defined iteration length. Some teams work continuously using short time-frames, usually shorter than one week, or choose bigger iterations like quarterly goals.

2.2.3.4 Kanban Artifact

The only artifact defined in Kanban is the *Kanban board* which is used to visualize work, limit WIP, and maximize the efficiency of the workflow. A basic Kanban board structure includes:

- *Column*: it is a specific process step, for instance *to do*, *doing* and *done*;
- *Visual Card*: it is used to write one work item, it can sometimes encapsulate one user story;
- *WIP Limits*: it is a maximum number of cards that can be present in one column at any given time. WIP limits allow controlling the workflow and giving the warning sign when there are too much of the committed works which have not been done;
- *Commitment Point*: it is when an idea is picked up by the team and work starts on the project;
- *Delivery Point*: it is the moment that the product or service is in the hands of the customer. In Kanban, the main goal is to take cards from the commitment point to the delivery point in the shortest time.

2.2.4 Scrumban

Scrumban is a mixed methodology of both Scrum and Kanban which increases adaptability and universality for product manufacturing and support focused companies [126]. According to Baleviciute [15], Scrumban allows saving time by

using *planning on demand* technique where the team plans only when there is a demand and there is no estimating or sprint planning needed. By saving time on planning, the team can put effort into quality control and verify if the work item is ill-formed. It also allows controlling a manufacturing process and to inspect if work is promoted to the ready queue. Scrumban focuses on waste minimization by using inter-process buffers and flow diagrams to show the weaknesses and opportunities of the process. This methodology allows eliminating everything that is not adding value to the customer.

As a mix of Scrum and Kanban, this methodology respects the same values and principles of the two methodologies.

2.2.4.1 Scrumban Roles

- *Product Owner*: this role is responsible for maintaining the product backlog by representing the interests of the stakeholders, ensuring the value of the work the Development Team does;
- *Scrumban Sensei*: this role is responsible for the correct use of the Scrumban process. Although the designation of a Scrumban sensei and its presence in Scrumban meetings are generally advisable, teams with a lot of Scrumban experience may also work without this role;
- *Development Team*: it is a cross-functional group of people who are responsible for delivering potentially shippable increments of the product at the end of every production cycle;
- *Stakeholders*: it is the people who enable the project. They directly involve in the process only during the reviews. Apart from that, they may solely influence the team by discussing their needs with the product owner. Typically, the main stakeholders are managers, customers, and users.

2.2.4.2 Scrumban Artifacts

- *Product Backlog*: it is an ordered list of requirements that the team maintains for a product. In Scrumban, one should document requirements in "user story" format. Anyone can edit the backlog, but the product owner is ultimately responsible for ordering the user stories. Stories in the product backlog contain rough estimates of both business value and development effort;
- *Selected Backlog*: it is a list of work the Development Team must address next with a defined capacity limit (also known as work-in-progress). As soon as capacity is available, it is filled up with user stories/features from the top of the product backlog;
- *Story In Progress (SIP) Backlog*: it is a list of user stories, which the Development Team currently addresses. Team members pull user stories from the selected backlog when there are no more remaining tasks in the task backlog;

- *Task Backlog*: it is a table structured along with the phases that are necessary for completing the project, e.g. design, development, and test. The Development Team breaks the user stories/features from the SiP backlog down into single tasks. Once a task has finished one phase, a team member from the consecutive phase eventually pulls the task to process it further;
- *User Story*: it is a description of a certain product feature or behavior, written strictly from the user's point of view. It is usually the product owner who writes the user stories;
- *Task*: it is a unit of work which should be feasible within one working day or less. To implement a user story, a team must accomplish all associated tasks;
- *Parking Lot*: it is for the tasks which the team cannot finish due to external dependencies. For example, another team has to review a document. Placing a task in the parking lot prevents the team from deadlocks, where unfinished tasks block production lines;
- *Cumulative Flow Diagram (CFD)*: it is a publicly displayed chart showing a detailed view of the teams' past and present performance. The CFD allows identifying bottlenecks in the production flow. It also enables the product owner to predict the time a new requirement will most probably need to complete;
- *Impediment Backlog*: it is a list maintained by the sensei, including all current impediments.

2.2.4.3 Scrumban Practices

- *Extend Board*: on the board, team create columns which consist of "To do", "Work in progress" and "Done". Column "Work in progress" can then be divided into more columns to indicate the particular stage a task goes through. Using these columns, everyone knows the current situation, and tasks are completed as soon as possible. New tasks are put on the board without assigning them to a particular team member. For this reason, team members can choose which task they would like to work with;
- *Backlog Limit*: team makes a list of tasks, puts them into the backlog, and sets work in progress limit for this column. Because Scrumban does not have regular planning meetings, a limit is used to implement the planning on the demanding technique. The team pulls items from the backlog into the process until it becomes empty, and an empty backlog is a trigger to notify that it is time to plan more tasks. It is better and easier to plan small, aiming only a few tasks per iteration. Team can also use the prioritization on the demanding technique which provides the team with information on which task must be taken next;

- *Find Bottlenecks*: divide team's work in progress section into smaller columns to implement separate WIP limits. This approach allows the team to discover bottlenecks which block the process flow;
- *Metrics Performance*: Scrumban uses the average lead and cycle time as its key metrics for performance. If lead and cycle time is under control, then the team can understand how long does it take for a task to reach the end consumer, how long it takes to develop and how long does it take to manage management. With these metrics, team can predict how long it will take to provide a certain amount of value or earn some amount of money.

2.2.5 Lean Software Development (LSD)

The Lean Software Development (LSD) comes from the Lean manufacturing of Toyota production system and Charette's Lean development in the 1980s [62]. In 2003, Lean was defined as software development by Poppendieck and Poppoendieck [116]. According to the authors, LSD is an iterative methodology that focuses on minimizing waste while maximizing customer value through the optimization of the entire process. LSD is considered to be more of a philosophy which has no specific role or artifact but rather emphasized project management. It was largely used in manufacturing and recently in software development to a lesser extent. LSD is ideal for projects where there is a need for radical change. The ultimate goal of a lean organization is to provide *perfect value* to its customer with *zero waste*. It does so by focusing on its key processes and continuously improving them. The optimization of each process is the core idea of the Lean philosophy. Instead of practice,

2.2.5.1 Lean Principles and Thinking-Tools

Poppendieck and Poppoendieck [116] promote *seven* principles in SLD. The methodology revolves around these principles, and all other aspects of Lean are designed to reinforce them. The authors also introduced 22 thinking tools which allow achieving the **seven** SLD principles and can help the team customize the right agile practices for any environment. We describe the principles and their supporting thinking tool as follows:

- *Eliminate Waste:* Eliminate anything that does not add customer value.
  - Seeing Waste: seven types of manufacturing waste translated into the software domain including partially done work, extra processes, extra feature, task switching, waiting, motion and defect;
  - Value Stream Mapping: it is a good way to discover waste in the process. This involves drawing a chart of the average customer request, from arrival to completion. At the bottom, draw a timeline that shows how much time the request spends in value-adding, waiting, and non-value-adding activities.

- *Amplify learning:* Use short iterative cycles to provide quick, constant feedback to ensure the right things are being focused on.
  - Feedback: increasing feedback is the single most effective way to deal with troubled software projects. Developers should know their immediate customer and have ways for that customer to provide feedback;
  - Iterations: iterations provide a dramatic increase in feedback. Iterations are a point of synchronization between the different teams and the customer. Iterations force decisions to be made because the system is deployed early and often;
  - Synchronization: build the system every day after a small batch of work has been completed by each of the developers and followed by an automated set of tests;
  - Set-Based Development: it starts by defining everyone's constraints and then selects a choice that fits into those constraints. Talking about constraints allows developers to defer making choices until the last possible moment.
- *Decide as late as possible*: do not make decisions until enough is known to make the decision—a sound understanding of the problem and the trade-offs of potential solutions is required.
  - Options Thinking: agile processes create options that allow decisions to be delayed until the customer needs are more clearly understood and the evolving technologies have had time to mature. Delaying irreversible decisions leads to better decisions, limits risk, helps manage complexity, reduces waste, and makes customers happy;
  - The Last Responsible Moment: it is the moment in which failing to make a decision eliminates an important alternative. If commitments are delayed beyond this moment, then decisions are made by default, which is generally not a good approach to making decisions;
  - Making Decisions: there are four types of decision making. Breadth-first involves delaying decisions. Depth-first involves making early commitments. Intuitive decision-making relies on past experiences rather than rational thought to make decisions. Rational decision making involves decomposing a problem, removing the context, applying analytical techniques, and exposing the process and results for discussion;
  - Pull Systems: the set of user stories is not assigned to developers; the developers choose the feature they want to work on.
- *Deliver as fast as possible*: minimize the time it takes to identify a business problem and deliver a system or feature that addresses it.
  - Queueing Theory: it strives to make the wait as short as possible. The fundamental measurement of a queue is cycle time which can

be reduced by controlling the rate of work arrival and removing the variability in the processing time;

- Cost of Delay: give the team an economic model that will empower the members to figure out for themselves what is important for the business.

- *Empower the team*: empower the team to succeed involving developers in the details of technical decisions is fundamental to achieving excellence.

  - Self-Determination: create an environment in which capable workers can actively participate in running and improving their own work areas;
  - Motivation: it starts with a clear and compelling purpose. After that, empower the team by ensuring that the purpose is achievable, giving the team access to customers, letting the team makes its own commitments, using management's role to run interference, creating a sense of belonging, providing a safe environment, and encouraging the desire to make progress;
  - Leadership: a successful team should have a respected leader who is excited and passionate about their work, exceptional developers who exercise leadership through superior knowledge, and a project manager who can identify waste, coordinate iteration planning meetings, help the team acquire resources, coordinate/ synchronize multiple teams, and provide a motivating environment;
  - Expertise: share Expertise by promoting mentorship and pair programming. Also, enforce standards in the development such as naming standards, coding standards, language standards, checking-in/out standards, and building standards.

- *Build integrity in*: system integrity comes from wise leadership, relevant expertise, effective communication, and healthy discipline.

  - Perceived Integrity: to prevent developers from getting lost in the details and customer value, visions of perceived integrity should be refreshed regularly through customer feedback. Simultaneously, the team needs to construct domain models such that software implementation can flow directly from these models which can be understood and directly usable by the customers and developers;
  - Conceptual Integrity: it is measured by how well a system's components work together as a smooth and cohesive whole. It can be achieved by effective communication;
  - Refactoring: complex systems have effects that are not fully understood at design time. However, the architecture must remain healthy as the system evolves. It is also important to maintain conceptual integrity by simplicity, clarity, suitability for use, no repetition, and no extra features;

Table 2.4 Mapping LSD principles with Agile Principle

Agile Manifesto	LSD
Principles	
- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software	Deliver as fast as possible, Build integrity in
- Deliver working software frequently	Deliver as fast as possible
- Business people and developers must work together daily throughout the project	Build integrity in
- Build projects around motivated individuals	Empower the team
- The most efficient and effective methodology for conveying information to and within a Development Team is face-to-face conversation	Build integrity in
- Continuous attention to technical excellence and good design enhances agility	Build integrity in, See the whole
- Simplicity, the art of maximizing the amount of work not done is essential	Eliminate waste
- The best architectures, requirements, and designs emerge from self-organizing teams	Empower the team
- At regular intervals, the team reflects on how to become more effective	Amplify learning

  - Testing: it proves that design intent is achieved and that the system does what customers want it to do. Tests should be automated as much as possible and run as part of the daily build.

- *See the whole*: use cross-functional teams to keep from missing important, possibly critical aspects of the problem and of the system designed to solve it.

  - Measurements: measurements are important for tracking the progress of software development. Try to create measurements that will measure everything such as standardize, specify and decompose;
  - Contracts: a common misconception is that agile development cannot be used in the context of contract work in which each firm is expected to look out for itself. We can incorporate LDS with different types of contracts, including fixed-price contracts, time-and-materials contracts, multistage contracts, target-cost contracts, target-schedule contracts, and share-benefit contracts.

Table 2.4 lists the results from mapping LSD principles with the Agile Principle.

2.3 Conclusion

In this chapter, we present the state-of-the-art of agile methodologies. The common ground shared by most of the agile methodologies was defined by four

agile values and twelve agile principles, also known as the “Agile Manifesto”. Over the years, a lot of agile methodologies in the practice have been proposed. Based on the survey, the most popular methodologies nowadays are Scrum, eXtreme Programming, Kanban, Scrumban, Lean, and the combination of these methodologies. Even though different agile methodologies define their own set of values and principles which fit different contexts and environments, most of them are still covered by the Agile Manifesto. In the words, the Agile Manifesto should have a perceptible influence on the values and principles of most of those methodologies. As agile practices are created in order to achieve the values and principles of each methodology, the relation between the Agile Manifesto and agile practices should thus be further studied.

Chapter 3

Agile Methods Tailoring: an Overview

In practical software development, simply choosing a particular agile method and following every rule will be tedious. To minimize efforts, the team should only adopt the most suitable methods or practices that best fit their situations. It helps to eliminate unnecessary efforts and avoid failure. The selection that makes a method more adherent to the development context is known as *software methods tailoring* [61].

This chapter provides a review of different approaches for software methods tailoring in general and for agile methods. This chapter is structured as follows. First, we discuss general software methods tailoring including contingency factors and methods engineering in Section 3.1. We then continue to discuss agile methods tailoring in Section 3.2. Finally, we make a conclusion in Section 3.3.

3.1 Software methods tailoring

Software development methods have been defined with the assumption that they can be applied to any type of project and application development [98]. Empirical research shows that methods used in software practice are rather limited, and those developers who use the methods tend to use different combinations and parts of methods rather than following all the steps required by a particular method [61]. With numerous projects failure, the software development community started to recognize the problem and search for a way to tailor the process to be more adherent to the context of the development. Software method tailoring has been discussed for a long time and continues to be a current need [41]. No matter the selected method or process, tailoring is normally needed for context adequacy in either organization or project levels [78]. Tailoring can be supported and stimulated by the adopted software method or executed based on the organization's needs [61].

Tailoring in software process context can be defined as the adaptation of the method to the aspects, culture, objectives, environment, and reality of the organization adopting it [31]. The options to execute software method tailoring have been studied and can be mainly classified as *contingency factors* and *method engineering* [60].

3.1.1 Contingency Factors

Research on contingency factors has been a long and continuing research stream in the information system. Davis [45] is one of the early and widely cited contributions. The author proposed the concept of strategy selection based on uncertainties concerning information requirements determination processes. In this approach, an appropriate strategy is selected from available alternatives based on an assessment of different levels of uncertainty. Thus, in Davis' model, an organization would be expected to have several alternative methods available and the developers would be expected to be highly experienced with each method to select the most appropriate one for the right situation. Similarly, Gremillion and Pyburn [69] also proposed a contingency approach where the development projects are evaluated according to the criteria of commonality, impact, and structure before deciding on a development approach. Avison and Wood-Harper [13] reviewed various Information System Development(ISD) methods and concluded that none can be appropriate in all situations. They then proposed their contingency framework called Multiview. This framework has been devised which includes descriptions of relevant techniques and tools. The analysts and users select those aspects of the approach which are appropriate to the context, in effect creating a unique methodology for each application. Benyon and Skidmore [19] proposed another approach which is known to be a single tool kit. In this toolkit, they combined the essential features of five different approaches ranging from soft to hard, and from process-driven to data-driven. To use this approach, developers would be expected to be skillful enough to choose the appropriate method or tool depending on the situation.

Based on the above-mentioned researches, Contingency research is typically premised on the notion that specific features of the development context are mapped to the selection of an appropriate ISD method from a portfolio of methods. The assumption is that there is no software development method good enough for the cases an organization can face when developing software [41]. The contingency factors approach handles the tailoring of software development methods by choosing multiple methods at once for the organization. This selection is generally based on development context features such as uncertainty level, impact, and structure.

The main challenge for contingency factors adoption is that the team members would have to understand and be capable of executing a range of methods according to the contingencies needed for the context. The contingency approach was criticized by Kumar and Welke [89] for being inadequate to cover all contingencies, and further, that the cost of sourcing and training for each method that is required by the contingencies of development. According to the authors, this situation would be further exacerbated by the rapid and fundamental changes in the prevailing development environment in organizations. In addition, the contingency literature does not provide enough practical guidelines on how methods or tools may be mapped to development contingencies. The solution proposed by Kumar and Welke [89] is "method engineering".

3.1.2 Method Engineering Theory

While ISD methods in their provision of a disciplined standard for development give a lot of advantages, it still requires flexibility so that the methods can be tuned to meet specific project needs Harmesen et al. [70]. To harmonize the methods, they propose a method-based repository that contains suitable method fragments and build situational methods out of the existing method fragments. The authors provide a detailed description of the application of situational method engineering, but the example they used to illustrate their approach is drawn from a literature example rather than a real case study.

According to Tolvanen and Juha-Pekka [155], methods should be constructed according to the needs of particular ISD situations and contingencies. To develop ISD methods and improve their flexibility, the authors developed methodical guidelines that are founded on engineering principles. In the guideline, they specified how knowledge related to methods should be described, analyzed, and maintained for ISD projects, and how it should be adapted into ISD tools. They thus built a meta-model which they defined as a modeling process that takes place one level of abstraction and logic higher than the standard modeling process. Their meta-model captures information about the concepts, representation forms, and uses of a method. The author also suggested that meta-modeling provides advantages in terms of representing, systematizing, and comparing methods.

Based on these previous researches, Method Engineering theory is a meta-method process and the creation of a new method to be applied to specific contexts using the existing method fragments. It is the creation of organization or project-specific methods and not the acquisition of existing methods from the community or a vendor [72]. The purpose is to build a more efficient method responding to the challenges of actual projects and context. While it brings flexibility to the owners, it however introduces challenges such as how to control the fragments or how to assemble the method for the context-specific situations [72]. One common feature of both the contingency and method engineering research is that they are largely deductive in nature. They employ theoretical and conceptual arguments to support how methods should be tailored or constructed. Very little information is available in terms of practical applications of these ideas in real life.

3.2 Agile methods tailoring

Like other software development methodologies, Agile methodologies are not always used to their full extent due to some constraints. Instead, software companies adopt agile methods in different ways based on different criteria such as business goals, culture and resources [5]. According to Kurapati et al. [90], practices selection allows organizations to best achieve their goals. Campanelli et al. [32] did an SLR on 56 research papers on agile methods tailoring. The authors then defined 6 types of research work on agile methods tailoring including most used practices, quality, business goals, maturity model, agile values, and project types. In addition to these researches, there is another direction, namely “created meta-models”. This research branch aims to formalize

the descriptions of agile methods and to make the method adoption process better, faster, and lower risks [43, 105, 139, 97].

3.2.1 Most used practices

Jalali and Wohlin [76] presented a list of practices found from an SLR on the use of agile methods in global software engineering (GSE). They also identified different conditions and factors, which affect the success of agile practices in GSE contexts. By analyzing 77 peer-reviewed papers and articles, 25 practices were identified as the most frequently used. These practices have been described to be successfully adopted based on different factors including project size, duration, domain, and the knowledge area.

Kurapati et al. [90] wanted to understand which agile practices are used in industry. They did a survey with 109 participants and analyzed practice adoption associated with the project and organizational level. Twenty-five practices from two agile methods (XP and Scrum), were presented to the participant to choose from. The results of this survey help us to identify the most commonly adopted practices. It also shows practices that are generally adopted together and correlated customer satisfaction after adopting the practices.

3.2.2 Quality

According to de Azevedo Santos et al. [46], organizations adopt agile practices as a means to achieve quality in their product. To validate their premise, they conducted a quantitative survey with 109 participants working on different fronts of the process of software and analyze the impact of agile practices on quality. In other words, their work focused on the quality aspects of the agile practices associated with the high quality of the software product. The survey results show that from the perspective of using agile practices, the quality of the software product can be improved by three aspects: bigger involvement of the staff, agile management of the requirements proposed, and code developed.

3.2.3 Business goal

Esfahani et al. [57] proposed a knowledge-based framework called Strategic Analysis for Agile Practices (SAAP). The objective of the framework is to associate the business goals of the organization with the agile practice selection process using concepts from the Balanced Score Card (BSC). In this framework, organizational strategies are significant situational attributes that affect the choice of agile practice. The framework consists of three main components, including the Strategies Graph, the Evidential Knowledge Base of Agile Practices, and the Strategic Analysis Process. First, important strategic goals of the organization are extracted, classified, and visualized. Then, the strategic knowledge about how each agile practice contributes to different strategic goals under various project conditions is retrieved from empirical studies. To situationally analyze the strategic impacts of every candidate agile practices, they use Strategic Graph.

3.2.4 Maturity model

To adopt agile practices based on an agile maturity model, Sidky and Ahmed [141] introduced Sidky Agile Measurement Index (SAMI) framework derived from agile values and principles. It was designed to guide organizations seeking to become more agile. The goal is not to focus on any particular methodology, but rather embracing and realizing the actual values that make a team Agile. There are 5 steps in the framework, including collaborative, evolutionary, integrated, adaptive, and encompassing. Each step aims to instill a new value in teams and organizations. To use this framework, Agile Coach creates an instance of the SAMI for a particular team by populating each of the steps with a set of practices that help the team or organization embrace the value designated by that step. A large number of these practices can come from existing agile methods or an in-house set of practices that suits their specific environment and constraints.

3.2.5 Agile values

According to Mardi et al. [100] agile values such as quickness, flexibility, and responsiveness are the reason behind the fame of agile methods. These values are fundamental that define the culture of the software company where a set of practices can be followed based on them. Their research focused on understanding the key agile values and how frequently they were mentioned in the literature. They started by looking into the experiences published previously in the agile literature. They then collected and analyzed how the agile practitioners identify agile values. Finally, they followed up on the analysis by collecting and comparing these values with the comments of Agile Manifesto signatories[1]. The agile key values obtained from this work include flexibility, customer-centric, working software, collaboration, simplicity, communication, natural, learning, pragmatism, and adaptability.

3.2.6 Project

Saleh [129] provides a methodology to select the best agile practices for projects based on the association between the project's characteristics and the abilities of the agile practices. The key project areas which were analyzed in the work include team size, iteration duration, and team distribution. In his model, first, the abilities of agile practices are classified based on a detailed analysis of existing research in the field of agile development. The practices are then clustered according to their abilities using K-means clustering. After that, a model of the Agile System Development Life Cycle (Agile SDLC) phases is used to categorize the practices based on their role in the development process. Finally, rules are applied to match project characteristics with the abilities of practices and produce a list of recommended practices.

[1] https://agilemanifesto.org/display/index.html

3.2.7 Meta-model for agile method tailoring

The meta-model for agile method tailoring aims at making relations, attributes, control flows, rules of a particular process more appealing based on a particular perspective, e.g., products and capability assessment [71], partial agile adoption [105], and goal-oriented [97, 139, 56].

Damianil et al. [43] proposed a meta-model that supports the derivation of specific data models for agile development processes. They defined two meta-models based on SPEM (Software Process Engineering Meta-model) specification [110]. One is used to generate models for describing the development process while the other is used to generate models for describing a measurement framework. To connect the two meta-models they defined a simple trigger layer. Their work provides the basis for a framework to model a generic software process meta-model and related measures which they claimed to be derivable into a model of any specific agile methods such as Scrum, XP, etc.

Mikulnas et al. [105] proposed a meta-model which enables the fusion of different agile methods referred as *partial agile methods adoption*. The core idea is to decompose each agile method into components which in theory could always be categorized into a common pre-defined structure. To create this meta-model, they analyzed the problem of a partial agile method adaptation and decomposed them into sub-problems. They then defined a set of concepts, where each of them has a direct or indirect relationship to the sub-problems. The integral solution has been achieved by developing these concepts, deriving their classes, and organizing them into the framework of the partial agile method adaptation. The constructed meta-model for the framework serves as a structure for the decomposition of the agile methods. It is a guide for creating patterns and developing models for the partial implementation of the agile methods from these patterns. Their work offers great flexibility in adopting agile methods since one can manually select and combine different alternative components coming from different methods at will.

To complement existing process modeling approaches with a new perspective, Esfahani et al. [56] proposed another type of modeling aimed at describing and analyzing the social and human aspects of a software process. In their work, i* modeling framework [54] is used to model the social perspective in agile methods. Their framework is composed of two processes. First, they use a Strategic Dependency (SD) model to depict the dependencies between different (social) actors. Then, they refine the SD model with a Strategic Rationale model. The SR model is used to illustrate actors' internal goals and the combination of activities, artifacts, qualitative attributes, sub-goals, and dependencies that help the actors to achieve their major goals. An explicit representation of the social requirements enables software companies to assess the chances of success by highlighting the major vulnerabilities of the process, checking (before the adoption process whether the social aspects of the process will be a good fit for current team members, and answer different kinds of social/organizational questions about the process.

Lin et al. [96] proposed a novel goal-oriented method to model a software development process on the top of the Agile Unified Process [10], called Goal Oriented Agile Unified Process (GOAUP) that could also be applied to OpenUP

and other agile-oriented forks of the unified process. In their model, the top goal for one iteration named *Software iteration finished* is a composite state. This goal is contributed by sub-goals where each of them is achieved by four lead transitions: inception → elaboration → construction → transition. The model of AUP via the Goal-Net method provides a new way to look at AUP from a goal perspective.

3.3 Conclusion

Software method needs to be tailored to be more adherent to the context of the development and to avoid the failure of the software development. The options to execute the software method tailoring have been studied and can mainly be classified as *contingency factors* and *method engineering*. Contingency research is the notion that an appropriate ISD method from a portfolio of methods is selected based on specific features of the development context. Method engineering theory is the creation of a new method to be applied to specific contexts using the existing method fragments.

In agile methods, there are 6 groups of researches on agile methods tailoring including most used practices, quality, business goals, maturity model, agile values, and project types. Researches about *most used practices* discuss how most popular agile practices are chosen based on different conditions and the factors which affect the success of agile practices. In another research group, *quality* in the software product was found to be the reason that organizations adopt agile practices. Other than that, *business goal* of the organization was associated with the selection process for agile practice adoption. There is research that provides a framework derived from agile values and core principles to help guide organizations seeking to become more agile-based on *maturity model*. Similarly, *agile values* have been discussed to be the reason behind the fame of agile methods. Finally, there is a group research that creates a methodology to select the best agile practices for projects based on the association between the project's characteristics and the abilities of the agile practices. To formalize their descriptions about agile methods and to make the method adoption process better and faster, and at the same time to minimize the risks, meta-models have been created from different perspectives.

Chapter 4

Socio-Intentional Modeling Framework: an Overview

Modeling is a one of the main activities in the software development process. Illustrating information in graphical models allows practitioners to understand and analyze agile practice easier. Goal-oriented modeling has been popularly used in the software engineering discipline with different definitions of the term "Goal". According to KAOS modeling [127], goals are desired system properties that have been expressed by some stakeholder(s). Based on Yu [67], goals describe the objectives that should be achieved through the cooperation of actors in the software-to-be and the environment. Illustrating the information about agile methods adoption in the goal perspective allows practitioners to identify the strategy to achieve their desired goals. As defined in the Agile Manifesto, agile methods focus on the social aspects more than the others including process, technique, tool, etc. In other words, the success of agile methods adoption depends highly on the individuals, their interactions and collaboration. It is thus important to find a modeling technique that can encompass human related issues in agile practice adoption. In socio-intentional modeling frameworks such as i* [165], Tropos [149], and iStar 2.0 [42], the role is one of the core elements of the modeling constructs. Each role is constructed with goals it wants to achieve and multiple dependencies to one another to collaboratively achieve their goals. Such a representation can help practitioners evaluate whether a practice is socially compatible with their team. It also allows reducing adoption failure by minimizing the vulnerabilities caused by roles.

In this chapter, we review some socio-intentional modelings frameworks that allow us to represent and analyze goals and dependencies between roles using various techniques. Reviewed frameworks are the popular ones that provide the most representation notions related to either goal or social dependency. In the following sections, from Section 4.1 to Section 4.4, we respectively describe KAOS, NFR framework, i* modeling framework, iStar 2.0, and Tropos. In Section 4.6, we provide a conclusion about these frameworks.

4.1 Knowledge Acquisition in autOmated Specification (KAOS)

KAOS is a methodology proposed by Dardenne et al. [44] for requirements engineering, enabling analysts to build requirements models and to derive requirements documents. A major benefit of KAOS resides in the fact that it provides a continuum between the problem description and the expected solution description. This bi-directional traceability between problem and solution spaces is fundamental not only for the requirements analyst to be sure that the system to build will be the right one but also for developers who need to understand the context and objectives to make correct architectural and design choices. The complete KAOS model leaves no space for wishful thinking (a goal unrefined), no space for requirements for which we do not know who is responsible, no space for unjustified operations, and no space for operations, for which we ignore who will execute what and when.

KAOS model explains step by step how to build a complete KAOS model and how to generate a requirement document using a supporting tool called *Objectiver* [127]. Their methodology to create a complete requirement document was proposed based on **ten** the following key ideas :

1. Build a requirements model for describing the problem to be solved and the constraints that must be fulfilled by any solution provider;
2. Justify your requirements by linking them to higher-level goals. A goal can be justified by the other goal that explains why it was introduced in the model or refined as a collection of sub-goals describing how the refined goal can be reached;
3. Build a model of the whole system, not just the software part of it. It is very important to identify, record, and take into account all the requirements and assumptions about that part of the environment that interacts with the software system;
4. Build a responsibility model which includes agents which can either be human or automated components that are responsible for achieving requirements and expectations;
5. Build a consistent and complete glossary of all the problem-related terms you use to write the requirements. In KAOS, analysts can work on the glossary progressively and simultaneously during goals and requirements definition by building a KAOS object model;
6. Describe how the agents need to behave in order to satisfy the requirements they are responsible for. KAOS provides *operation diagrams* which allows describing all the behaviors that agents need to have to fulfill their requirements;
7. Generate the requirements document based on the requirements model. Requirements on the system architecture are derived from the responsibility model and requirements for the system behavior from the operational model;

8. Validate your requirements by first reviewing the model is more efficient than asking people to read a long technical document;
9. Use a defensive approach to the building of a requirements model that investigates in a systematic way what can go wrong in the system-to-be, that is how and why some requirements can no longer be satisfied;
10. Consider your requirements document as a reference that shall need updating during

    the project development life cycle;

To support their methodology on the construction of a requirements model by means of diagrams, KAOS provides the graphical representation which can be described by the meta-model in Figure 4.1.

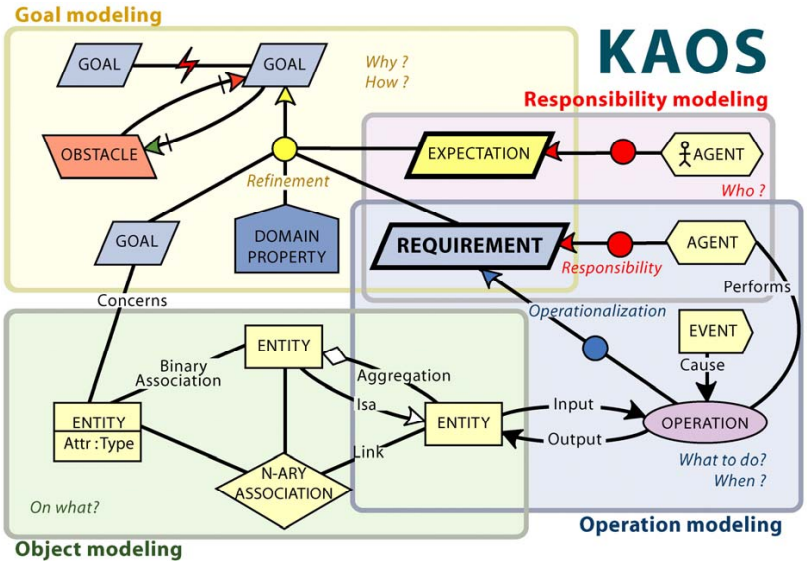


Fig. 4.1 KAOS meta-model (Objectiver [127])

4.2 Non-Functional-Requirement Framework (NFR)

NFR Framework was proposed by Chung et al. [37] aims to put the main focus on non-functional requirements in the developer's mind by using nonfunctional requirements such as security, accuracy, performance, and cost to drive the overall design process. The framework offers a structure for representing and recording the design and reasoning process in graphs, called soft-goal interdependency graphs (SICs), and cataloging of knowledge about NFRs and development techniques.

The purpose of this framework is to describe non-functional requirements which are shown as *soft-goals*. Starting from the top of a graph, *soft-goals* are

connected by *interdependency links* that describe the refinements of *parent* soft-goals downward into other *offspring* soft-goals. Each soft-goal has an associated label representing the degree to which a soft-goal is achieved, which shows the contribution of offspring soft-goals upwards upon the meeting of other *parent* soft-goals. The whole process design includes the decision of what soft-goals to state, how they are refined, what extent they are refined, and how to calculate the label value under the control of the developers.

Another important aspect of the framework is the possibility to draw the body of design knowledge (including development techniques) in three different kinds of organized *knowledge catalog*. The first kind of catalog represents knowledge about the particular types of NFRs and their associated concepts and terminology. The second kind is used to systematically organize available development techniques to help the developer to meet requirements. The third kind of catalog shows implicit interdependencies (correlations, trade-offs) among soft goals. These catalogs provide a valuable resource for use and re-use during the development of a variety of systems.

The design process consists of **eight** sequential and iterative steps as follows:

- Acquire or access the knowledge about the domain and the system being developed, functional requirements for the particular system, and particular kinds of NFRs, and associated development techniques;
- Identify the main non-functional requirements that the particular system under development should meet;
- Decompose NFRs soft-goals based on topic or NFR type;
- Prioritize soft-goals;
- Identify possible development techniques (operationalizations) for achieving the NFRs (soft-goals);
- Deal with ambiguities, tradeoffs, and priorities, and interdependencies among NFRs and operationalizations;
- Select operationalizations;
- Support decisions with design rationale;
- Evaluate the impact of decisions.

To support their methodology on how to describe non-functional requirements by means of diagrams, the NFR framework provides the graphical representation which can be described by the meta-model in Figure 4.2.

4.3 The i* Modeling Framework

Yu and Mylopoulos [165] introduced a modeling language called i* to highlight the importance of *WHY* dimension in software processes, to provide the intentional view of modeling domain which can express motivations, intents, and

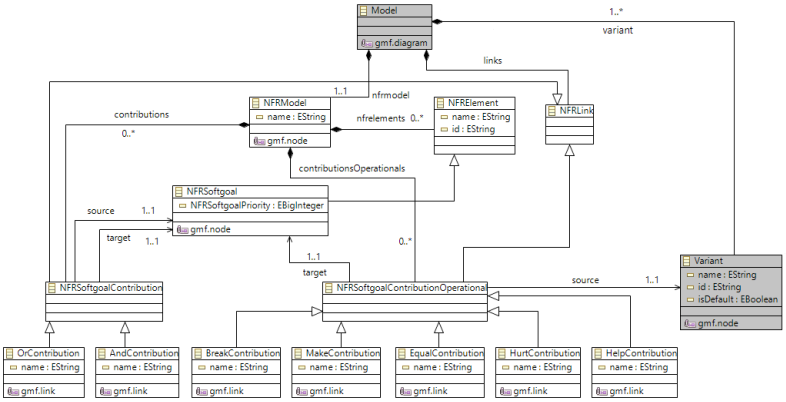


Fig. 4.2 NFR meta-model (Pereira et al. [115])

rationale behind activities. The i* language is presented as a goal and actor-oriented modeling and reasoning framework that consists of a modeling language along with reasoning techniques for analyzing created models. The i* language is composed of two modeling components: Strategic Rationale (SR) model for representing actor objectives and their alternative ways of fulfillment, also the required resources; and Strategic Dependency (SD) models for describing the dependency relations among organizational actors.

To build an SD model, i* proposes a set of nodes to represent actors (Agent, Role, and Position) and a set of dependencies to describe how an actor (depender) depends on some other (dependee) in order to obtain some objective (dependum). In i*, the types of dependency are classified based on the object of dependency (Goal, Soft-goal, Task, and Resource). The model also defines three degrees of strength (open, committed, and critical) for the dependency according to the level of vulnerability of the depender when the dependee fails to accomplish the specified element.

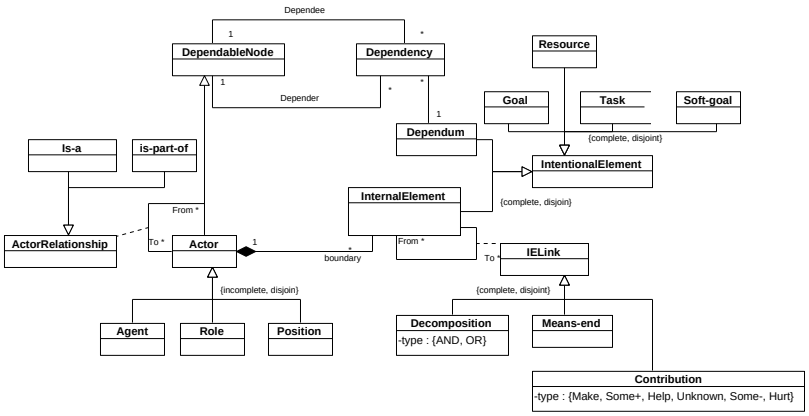


Fig. 4.3 i* language meta-model (Xavier et al. [67])

The representations of the dependency relations depicted in SD models are not as detailed as those represented in SR models. SR diagrams represent a more elaborate view of actors' intentional elements (Goal, Soft-goal, Task, and Resource). In order to do that, the intentional elements inside the SR model are refined accordingly using three types of links (1) *Means-end* links which indicates a relationship between an end, and a means for attaining it. The "means" is expressed in the form of a task, since the notion of task embodies how to do something, with the "end" is expressed as a goal. (2) *Task-decomposition* link to describe a task and its component. A task can be decomposed into different types of intentional elements. (3) *Contribution* link to describe how intentional elements can contribute (Make, Some+, Help, Unknown, Break, Some-, or Hurt) to the satisfaction of a soft-goal. Figure 4.3 shows a UML class diagram representing the i* language.

Based on the core concepts in i* language, two main frameworks, Tropos and iStar 2.0 have been introduced for a better representation and usage.

4.4 iStar 2.0

iStar 2.0 is an updated version of i* language, introduced by Dalpiaz et al. in 2016 [42]. According to the authors, there are some drawbacks of i* language that make it hard to be spread outside the experts' community. First, it is hard for newcomers to learn the intricacies of the language. Second, there is not any shared body of knowledge for the educator to teach. Thirdly, there is not any established reference for using i* in practitioners' projects, and finally, it is hard for technology providers to determine which are the core constructs to be implemented and the techniques to apply on top of those constructs.

The i* research community later started to identify a widely agreed-upon set of core concepts in the i* language to solve the aforementioned problems. To build a solid, unified basis for teaching and conducting research with i*, the community discussed this language in several meetings and discussions started from October 2014 until May 2016. The result of this two-year community effort is iStar 2.0 which is claimed to improve consolidation, simplicity, accuracy, usability, expressiveness, extensibility, and teachability. Figure 4.4 illustrates the meta-model of iStar 2.0.

There are obviously some differences between i* and the newest version iStar 2.0 which can be summarized in Table 4.1. Some concepts (nodes and links) in i* are omitted or replaced, and some new others are introduced in iStar 2.0.

iStar 2.0 also introduces another type of model view called *Hybrid* in addition to the existing model views (SD and SR) in i*. In this *Hybrid* view, the representation is flexible and can be defined accordingly. For instance, it can be used to combine SD/SR views where we want to focus on the strategic rationale of a particular set of actors, but not all. It can also be used to show only actors and actor links or to show only the functional view by hiding all the qualities, contribution and restriction links, etc.

Table 4.1 Comparison between i* and iStar 2.0 (Dalpiaz et al.[42])

Nodes and links	i*	iStar 2.0	Comment
Actors	General actors Roles, positions, agent	General actors Roles, agent	
Actor links	is-a is-part-of, plays, occupies, cover	is-a participated-in	iStar 2.0 simplify i* with a generic relationship that may be applied among two actors of many types
	INS	-	Agent can only be an instance in iStar 2.0
Intentional elements	Goal, task, resource	Goal, task, resource	
	Soft-goal	Quality	They move from hard/soft goal dichotomy as the distinction between the concepts is based on the level of satisfaction which is varied in practice
Intentional elements links	Means-end, task decomposition	Refinement	A single relationship for simplicity, different semantic depends on the connected elements and the logical connectors AND/OR
	Contribution	Contribution	
		Qualification, Neededby	New relationships to link goal/task to qualities/resource, respectively

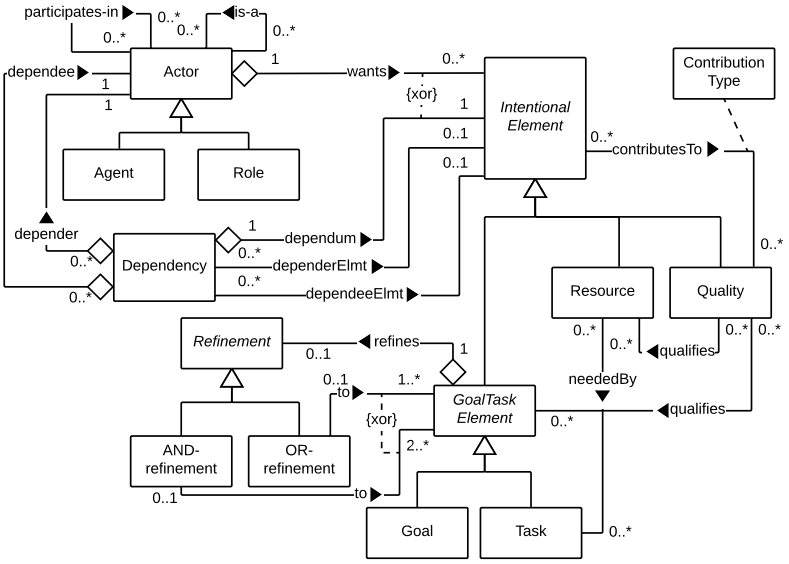


Fig. 4.4 iStar 2.0 meta-model (Dalpiaz et al.[42])

4.5 Tropos

Tropos is an agent-oriented software engineering (AOSE) methodology that covers the whole software development process proposed by Castro et al. [33]. Tropos proposed goal-oriented modeling techniques to capture and analyze both business and system requirements, risk and security, and (social and geographic) location requirements based on Eric Yu's i* modeling framework [164]. The Tropos methodology supports four phases of software development as follows:

1. *Early requirements:* It is the phase that the organizational model is created. This model is concerned with the understanding of a problem by studying the organizational context within which the system-to-be will eventually function. This model includes stakeholders, relevant actors, their respective goals, and how they depend on each other to fulfill their goals, to perform their plans, and to provide the resources. The outcome of this *early requirements* phase includes two main diagrams: the actor diagram and the goal diagram where the latter is a refinement of the former with emphasis on the goals of a single actor;

2. *Late requirements:* It is the phase where the system-to-be is described with a definition of the functional and non-functional requirements of the system-to-be within its operational environment. To do that, the system-to-be is treated as one actor which has a number of dependencies (can be a depender or a dependee) with the other actors of the organization. The requirement state changes from "early" to "late" when these dependencies are defined;

3. *Architectural design*: it is the phase where the system's global architecture is defined in terms of subsystems, interconnected through data, control, and other dependencies. Since the fundamental concepts of Tropos architectures are intentional and social, subsystems and system components are represented as actors and their dependencies to other system components are social, rather than procedural/structural. By this mean, system components need to have the ability to monitor dependencies to other actors and if any dependencies cannot be fulfilled, system components need to be able to cancel the dependencies and replace them with new ones, through planning, negotiation, etc.;

4. *Detailed design*: it is the phrase where the specification of actor communication and behavior is defined in further detail. Agents' goals, beliefs, and capabilities are specified in detail, along with the interaction between them.

As Tropos is a methodology which was proposed based on i*, their concepts and relationships are mostly the same. The primary difference of the Tropos goal model with i* is the possibility to decompose goals (AND/OR) in Tropos. The concepts related to the goal diagram in the Tropos meta-model are illustrated in Figure 4.5.

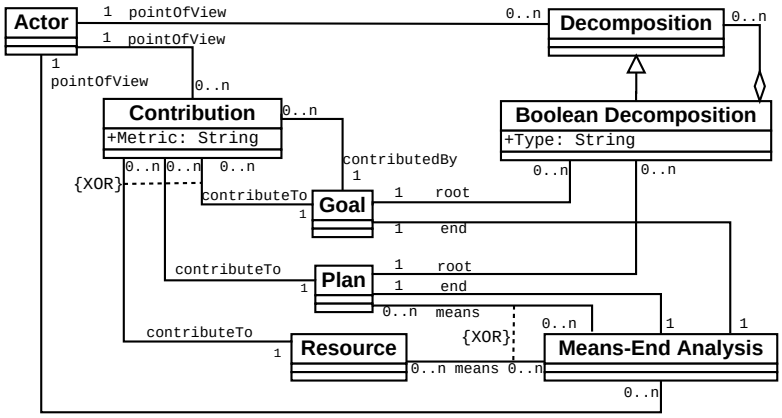


Fig. 4.5 Tropos meta-model for the concepts related to the goal diagram (Susi et al. [149])

4.6 Conclusion

Socio-intenional modeling allows us to fill the gap in the spectrum of conceptual modeling languages by focusing on the intentional (why) and social (who) dimensions. The analysis that focuses on socio-intentional aspects allows us to understand the dependencies between various team members and to clarify the motivations, intents, and rationale behind their activities.

In this chapter, we have reviewed five modeling frameworks considered to be the most well-known that can represent either goal or social dimension. Among them, 3 (KAOS, NFR, i*) were proposed with their own representations as well as the methodologies on how to build the model effectively. iStar 2.0 is the extended version of the i* modeling framework. Tropos is a methodology that explains how to use i* modeling framework to capture and analyze the requirement. Despite some similarities within these frameworks, they are used for emphasizing different aspects. KAOS provides a continuum between the problem description and the expected solution description. The methodology and representations in this framework allow describing the requirements of system-to-be, how to build it correctly, and who is responsible for each requirement. On the other hand, NFR focuses on how to describe non-functional requirements (also known as soft-goal) and identify techniques to achieve these soft-goals. Similar to the two previous frameworks, i* modeling also allows describing system requirements. In addition, i* provides extra representations and techniques that allow describing the dependency between actor, goal, resource, and task in order to achieve goals. iStar 2.0 is the most updated version of i* which is claimed to improve consolidation, simplicity, accuracy, usability, expressiveness, extensibility, and teachability of the original version. At the same time, iStar 2.0 also introduces another model view called hybrid which gives a broader way to build the socio-intentional model. Finally, Tropos proposes a methodology based on the representation in i* modeling framework to capture business and system requirement that supports four phases of software development.

Part III

Socio-Intentional Framework for Agile Methods Tailoring

Chapter 5

Agile Manifesto and Practices Selection: a Systematic Literature Review

The fundamental core of an agile method should be well understood before deciding on which parts of the method to adopt. The quickest way to do so is by understanding the Agile Manifesto. We have observed however that, in practice, agile practitioners do not dedicate the necessary attention to the manifesto and directly aim at a specific agile method or practice. This has led us to question the role of the Agile Manifesto in tailored agile methods adoption.

This chapter aims at understanding and verifying the relation between the Agile Manifesto and agile practice selection. To this end, we carried out a systematic literature review (SLR) to answer four research questions related to tailored agile method adoption i.e., the perception of the Agile Manifesto in adopting tailored agile methods, and what are the influences, relevance of the manifesto in meeting the team's problems, expectations, and benefits. This chapter is structured as follows. First, we explain the research context in this chapter in Section 5.1. We then discuss some related works in Section 5.2. Our research methodology, including details on research questions, search strategy, and data extraction is discussed in Section 5.3. Then, the results of our literature review are presented in Section 5.4 followed by the threads to validity in Section 5.5. In Section 5.6, we address the limitations of this chapter. Finally, our conclusion and findings are summarized and discussed in Section 5.7.

The content of this chapter is mainly taken from our article published in the *International Conference on Product-Focused Software Process Improvement (PROFES - 2018) proceeding* [84].

5.1 Introduction

Representatives from eXtreme Programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Adaptive Software Development (ASD), Crystal, Feature-Driven Development (FDD), and Pragmatic Programming met in 2001 to discuss and establish common ground for an alternative to structured and traditional heavy software development life cycles. They eventually emerged with a *manifesto* for Agile Software Development, commonly known as the

Agile Manifesto (http://agilemanifesto.org/), defining values and principles to be respected to be defined as *agile*.

No method can, of course, be a one-size-fits-all solution. Likewise, simply choosing a particular agile method and following every rule is also inconsistent and inefficient. Instead, software Development Team apply agile methods differently, i.e., depending on their problems, resources, and goals or expectation [5]. For instance, the Development Team will choose to adopt concepts and building blocks that are the most suitable to them based on their specific situation, goals, problems, constraints, etc. This selection makes the method more adherent to the development context; it is known as software methods tailoring [61].

Choosing agile concepts, or more concretely agile practices, to adopt requires sufficient knowledge of the concepts and the impacts these could have on the team. Understanding all the details of agile concepts could be a time-consuming and complicated task so that many approaches have been proposed in order to simplify agile methods tailoring.

One of the interesting topics in agile methods tailoring is the relation and straight-forward interpretation between each agile practice and agile values or principles [9, 18, 77, 83, 94, 100, 138]. On this basis, different ideas for agile methods tailoring have been suggested. For instance, Ahmed and Sidky [9] proposed the road-map to adopt agile practices based on five values, considered as the most essential to agility. According to Madi et al. [100], knowing the most important values is the key to follow the best set of practices as agile values are fundamental. They analyzed papers and books to explore the key agile values and the relationships between them. Our previous works [83, 86] illustrated the strong relationship between the Agile Manifesto (values and principles) and agile practices, together with an approach for practices selection using an intentional modeling framework.

Even though the ideas seem so rational and reliable, to the best of our knowledge no formal verification on the relation between agile practices and values or principles has been performed yet. Their relations were based on the assumptions or beliefs of authors. Moreover, although their relations have been supported by many researchers, we have observed that agile practitioners do not seem to agree that Agile Manifesto is important for the adoption of the agile method. In many cases [12, 16, 26, 48, 140], Development Teams do not dedicate any effort to understanding any agile value or principle before adopting any agile method. They simply adopt the specific agile methods or practices which have been known as popular and follow the prescribed process. In other words, practitioners nowadays are "*doing agile*" more than "*being agile*". While these two concepts sound similar, they describe two different ways practitioners adopt agile methods. "*Doing agile*" means adopting the practices by focusing only on the process, tool, and technique without understanding the values and principles behind them. On the other hand, "*being agile*" means adopting the agile practices by comprising the internal processes of the individual, which reflect the personal being in the agile context. "*Being agile*" focuses on the way of thinking of an individual towards agile values and principles [51, 123]. This observation is corresponding to the claim that "*Agile is Dead*" raised by Dave

Thomas, one of the Agile Manifesto authors [154]. According to the author, the word "agile" has become meaningless as none of the things practitioners are doing are in the spirit of the Agile Manifesto and they no longer use an "agile mindset" as a guide to what is useful in their practice.

These reasons motivated us to study and verify, from a statistical point of view, the relation between the agile values, principles and practices in tailored agile methods adoption, by means of a systematic literature review. Indeed, value and principle are subjective concepts that vary greatly from one method to another. Gathering all the values and principles in literature and categorizing them would require enormous time and effort. We leave thus this question for future research. Also, we aim to ease the selection process, having a limited number of concepts would definitely be helpful and efficient. Consequently, we decided to focus on the fundamental 4 values and 12 principles defined in the Agile Manifesto.

In this chapter, we conducted a systematic literature review to extract key information from the case studies such as: (1) *How has the Agile Manifesto and its importance been discussed in tailored agile methods adoption?* And (2) *Can the Agile Manifesto and agile practices selection be related?* We believe that this study will help to enhance the value of the fundamental ideas of the Agile Manifesto and make its importance more obvious to the community.

5.2 Related Work

Over the last decade, many agile methods have been proposed based on the Agile Manifesto to meet specific requirements and situations. For instance, Scrum is proposed with the objective to put more focus on project management organization while XP is designed to be more responsive to customer requirement changes [105]. Although agile methods are flexible, they may not be easy to adopt. To ease the process, various meta-models have been proposed [56, 97, 105, 135, 139, 162, 163], serving as a road-map for agile adoption. We note, for instance, the situational method framework [135], development process [105], goal-oriented meta-model [56, 97], Agile Unified Process [10], Goal-Net theory [139], etc.

Another research direction focusing on selecting agile practices during adoption is agile methods tailoring [5, 9, 32, 57, 90, 100]. Campanelli and Perreiras [32] analyzed methodological and practical aspects of research on tailored agile methods and the criteria used for agile methods tailoring. Their results show that practice selection is based on the internal environment such as project type, communication, culture and management support, and objectives. Qumer and Henderson-Sellers [121] also acknowledged the impact of organizational culture and technical aspects. Abbas et al. [5], Esfahani et al. [57], Kurapati et al. [90] and Madi et al. [100] provided a formalized answer on how to select agile practices for tailored agile methods adoption but admitted that no final academic solution was found on practice selection in tailored agile methods adoption.

Alongside the aforementioned approaches that depend mainly on the business goals, the culture and the resources of the organization, there exists a new group

of methods based on agile values and principles [9, 83, 100]. Madi et al. [100] identified 10 key agile values and show how frequently they were mentioned in the literature. Their identified agile values are: *flexibility*, *customer-centric*, *working software*, *collaboration*, *simplicity*, *communication*, *natural*, *learning*, *pragmatism* and *adaptability*. According to them, these 10 values constitute the most important influence on practitioners in practice selection. The Sidky Agile Measurement Index (SAMI) [9] showed the adoption of agile practices based on an agile maturity model. SAMI is a 5-step road map to guide adopting teams based on five values considered essential to agility: (level 1) *enhancing communication and collaboration*; (level 2) *delivering software early and continuously*; (level 3) *developing high quality, working software in an efficient and integrated manner*; (level 4) *respond to change through multiple levels of feedback*; and (level 5) *establishing an environment to sustain agility*. SAMI is not based on any specific agile method such as XP, Scrum or Crystal, but instead, uses agile values and principles to define the path to agility. However, the framework was built just based on assumptions of the author as mentioned in [9]. Lee and Yong [94] also claimed that each agile practice should help accomplish agile principles in a method and can be grouped into management practices, software process practices, and software development practices. Similarly, we defined in [83, 86] the relation between agile value, principle and practice in the goal perspective where principle contributes to value and practice is used to achieve the principle. We also proposed a framework which can be used to help to select practices. In all these references, agile value and principle are seen, directly and indirectly, as the set of goals that the Development Team needs to achieve in order to be agile and practice is used to help them accomplish these goals.

Motivated by these methods, we strongly believe that there is a relation between agile value, principle and practice from a goal perspective. In other words, when it comes to selecting agile practice, by understanding the Agile Manifesto, practitioners should be able to effectively and quickly distinguish the outcome of different practices more easily. Although such idea has been confirmed by many researchers [9, 83, 94, 100], its usefulness in supporting a practitioner to select an agile practice remains unclear.

Many SLRs have been performed with respect to many different aspects in agile methods, from the general concept such as [6, 40, 55] to the specific topics like [32, 75, 132]. Among all, the more closely related to our work is [32], a systematic literature review of 56 research papers on agile methods tailoring. It provides detailed literature on agile methods tailoring and a deep understanding of how the researches on agile methods tailoring were conducted. The authors identified also the research community view on agile method tailoring, and the research gaps on the theme. The result, however, does not prove anything about the relation between agile value, principle and practice.

5.3 Research Methodology

In this chapter, we adopt an SLR approach [82] to study and verify the relationships between the Agile Manifesto and agile practices, in the context of tailored

agile methods adoption. An SLR allows us to adopt a formal and systematic approach to identify, select and synthesize recent literature relevant to our research questions [82]. It consists of defining (1) research questions, (2) search strategy, (3) study selection, (4) data extraction, and finally (5) data analysis. Each step will be explained hereafter. Figure 6.2 illustrates the process we have followed.

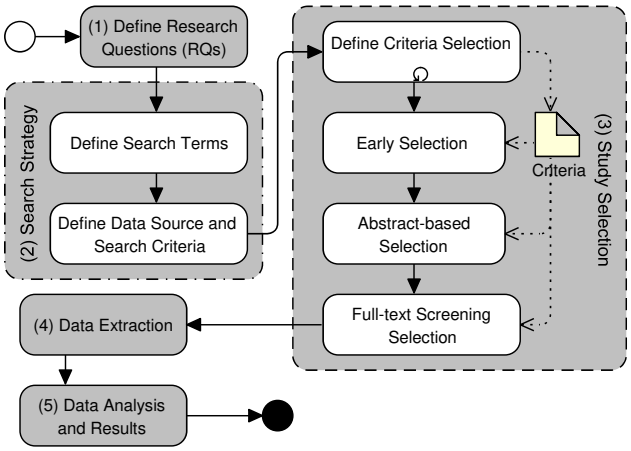


Fig. 5.1 Research protocol.

5.3.1 Research Questions

The main aim of this chapter is to confirm the relationship between the Agile Manifesto and agile practices, more specifically *whether or not the Agile Manifesto (i.e., the 4 values and 12 principles) is still the core concept that teams should understand before choosing an agile practice or method to adopt.* To answer this, we have formulated two fundamental Research Questions (RQs) in this chapter:

5.3.1.1 RQ1: How has the Agile Manifesto and its importance been discussed in tailored agile methods adoption?

This first question is to verify whether or not the Agile Manifesto has lost its attention and importance. Answering this question allows us to know about the state of the art of the Agile Manifesto from a practitioner's point of view; it includes:

- *RQ1.1: How often has the Agile Manifesto been discussed by agile practitioners during their adoption?*
- *RQ1.2: In which manner has the Agile Manifesto been discussed, as a whole or only part of it, just as a reminder or in detail?*

- *RQ1.3: Has the Agile Manifesto been recognized as important by practitioners for their adoption or not? If it has, how often and how has it been described?*

5.3.1.2 RQ2: Is the Agile Manifesto related to agile practices selection?

This question verifies whether or not there exist relations between the Agile Manifesto and practices, as mentioned by many researchers. As pointed out in the related work (see Section 5.2), agile values and principles have been regarded as a set of goals to achieve for a method to be agile. This set of goals is said to be accomplished by adopting agile practices.

We seek to answer this question by comparing the Development Team's goals of adopting agile methods with what is described in the Agile Manifesto. The results would allow us to confirm whether or not the Agile Manifesto could be related to agile practice selection, from the practitioner's point of view.

Based on our observation, Development Team's goals in adopting tailored agile methods can be described in three situations: (1) sometimes, Development Teams decide to change their development process based on *problems* they encountered. Their goal is to solve these problems by using a set of agile practices or methods; (2) in some other cases, *problems* are not the root cause of the adoption. Knowing that agile methods are the most popular nowadays, some Development Teams decide to follow them with the hope of improving their current processes. They have their predefined goals or *expectation* to achieve by adopting specific agile practices or methods; (3) regardless of the *problem* to solve or the *expectation* to fulfill, in many cases, result from adopting agile methods are described as *benefits*. These benefits can be seen as accomplished goals.

Hence, in order to know whether or not the Agile Manifesto could be related to agile practice selection, we defined three other sub research questions:

- *RQ2.1: Is the Agile Manifesto relevant to the team's problems that led to tailored agile methods adoption?*
- *RQ2.2: Is the Agile Manifesto relevant to the team's expectations from tailored agile methods adoption?*
- *RQ2.3: Is the Agile Manifesto relevant to the team's benefits of tailored agile methods adoption?*

If the Development Team's goals of agile methods adoption in most situations are relevant to the Agile Manifesto, then the description in the manifesto can still cover the core goals of this methods creation. Understanding the Agile Manifesto would allow the Development Team better defining their goals in adopting agile methods and consequently better selecting the set of agile practices.

5.3.2 Search Strategy

5.3.2.1 Search terms.

Our objective is to understand the importance of the Agile Manifesto in tailored agile methods adoption. Software method tailoring is the process that makes the method more adherent to the development context [61]. Various terms are used in the literature as a synonym for *tailor*, i.e., *partial*, *customize*, and *practice selection* [5, 14, 56, 83, 90]. Based on the research questions, we defined search terms as the combination of various words referring to tailored agile methods including *partial, tailor, customize, practice selection* and the name of the most popular agile methods (according to the 11th VersionOne survey [160]). Since we want to find out the *expectations/goals* of adopting agile methods, we thus also added the word "goal" into the search terms. We summarize the search terms as follows: "*(Agile OR Scrum OR XP OR Kanban OR ScrumBan OR Lean OR DSDM OR AgileUP OR FDD OR Iterative Development) AND ((practice AND select) OR tailor OR customize OR partial OR adopt OR expectation OR goal)*".

5.3.2.2 Search Engines and Search Criteria.

We only consider formal data sources, i.e., papers that were published in peer-reviewed conferences and journals from the four well-known digital libraries in the field of software engineering: IEEEXplorer (http://ieeexplore.ieee.org), ScienceDirect (http://sciencedirect.com), SpringerLink (http://link.springer.com) and ACM Digital Library (https://dl.a cm.org). We did not consider GoogleScholar since it provides also unpublished and non-peer-reviewed papers.

For each search engine, we used advanced search options to ensure our dataset quality. In general, we set the publication years between 2000 and 2017, the field of Software Engineering, and the search terms matching the title of the paper, keywords, or abstract. We found 13125 papers in total: 1722 papers in IEEE Xplorer, 526 papers in ScienceDirect, 9053 papers in SpringerLink, and 1824 papers in ACM Digital Library (see Figure 5.2).

5.3.3 Study Selection

We defined a 3-step paper selection process due to the number of papers found: *Early Selection*, *Abstract-based Selection* and *Full-text Screening Selection*. Each step, described in the following subsections, has well-defined selection criteria. Figure 5.2 provides the results of selected papers of each step.

5.3.3.1 Early Selection.

The goal was to have a consistent list of papers. All the search results were merged into a single file listing 13125 papers in total. We then eliminated redundant papers or papers not published in the 2000-2017 period. This step allowed us to discard about two-thirds of the papers to finally retain 4361 papers.

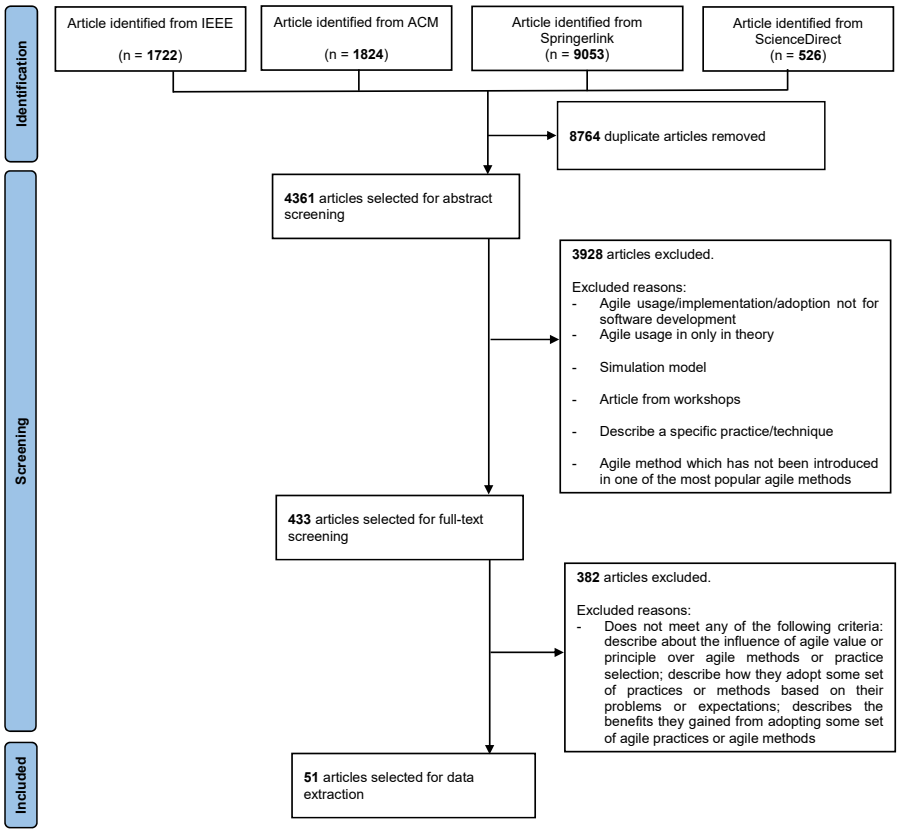


Fig. 5.2 Papers selection.

5.3.3.2 Abstract-based Selection.

The goal was to determine whether or not the article relates to our research questions based on its abstract which was carefully read by three reviewers. Before we started the real selection process, we defined and refined several times criteria for inclusion and exclusion to gather the maximum possible relevant articles and effectively reject irrelevant papers. The final criteria are summarized in Table 6.1:

We used Covidence (www.covidence.org), a collaborative tool for facilitating the SLR process.

To get started, we needed to upload the title and abstract of the 4361 papers into Covidence. However, since SpringerLink and the ACM Digital library do not allow downloading multiple abstracts at once, we therefore developed a third-party program for help. We then started the review process.

Each reviewer read the title and abstract of each paper and voted individually (Yes/No/Maybe) based on the above criteria. Papers with three 'Yes' votes were included for the next step, those with three 'No' votes were eliminated

Table 5.1 Inclusion and exclusion criteria for Abstract-based selection.

Inclusion criteria	Exclusion criteria
- Tailored/partial/customized agile methods or agile practices selection;	- Agile usage/implementation/ adoption not for software development;
- Empirical/research on adopting agile methods for software development;	- Agile usage in theory;
- Literature review/survey on agile framework;	- Simulation model;
- Challenge/issue in agile methods adoption;	- Article from workshops;
- Approach, model, framework, introduction or guide to agile methods adoption;	- Use of a specific practice/technique (daily meeting, pair programming, etc.);
- Integration of agile methods to other methods;	- Agile method which has not been introduced in one of the most popular agile methods
- Transformation from other to agile methods;	
- Agile practices usage.	

and papers with three 'Maybe' or conflicted votes were solved by a face-to-face discussion. 433 papers were selected for the next step.

5.3.3.3 Full-text Screening Selection.

The goal was to do a full-text screening of each paper and determine if it still relates to our research questions. We followed the same process as in the previous step and used the same tool.

First, we downloaded manually the full-text in PDF format and uploaded it to Covidence. 399 papers were successfully uploaded, and 4 papers were rejected for technical and format reasons. In addition to the abstract-based selection, we extended our inclusion criteria to the real case study which:

- describes the influence of agile value or principle over agile methods or practice selection;
- describes how they adopt some set of practices or methods based on their problems or expectations;
- describes the benefits they gained from adopting some set of agile practices or agile methods.

As long as one of the criteria is found, the article is included. In the end, 383 papers were eliminated and only 51 papers were selected in this study.

5.3.4 Data Extraction

Each paper was read carefully and data was extracted by only one reviewer. We divided the 51 papers into three sets and each reviewer took care of one set. For each paper, we extracted the following information:

- **Conference or Journal name and year of publication**: It allows us to determine if the dataset is representative for our study;
- **Type of agile**: It allows us to know in which environment the tailored agile methods are adopted;
- **Type of institution**: It allows us to know in which sector agile methods are tailored and adopted;
- **Mention about Agile Manifesto**: It allows us to answer RQ1.1 and RQ1.2. We denoted the findings as 'Yes' when the paper explicitly mentioned the word Agile Manifesto and we extracted values or principles and denoted them otherwise;
- **Agile Manifesto influence on partial agile adoption**: Basically, we tried to find a clear statement of influence by the authors. We denoted 'Yes' if the author simply refers to 'Agile Manifesto' as influential, or we extracted the values and principles if any of them were described as the influence. It allows us to answer RQ1.3;
- **Problem:** We read very carefully to understand the cause behind the agile methods tailoring. For any mentioned problem that led to agile practices or methods adoption, we extracted the specific statements without any modification and stored them in a list. Mapping this list to the 4 values and 12 principles of the Agile Manifesto allowed us to determine whether or not the Agile Manifesto is relevant to the team's problems and to answer RQ2.1;
- **Expectation**: We followed the same process for extracting problems. Instead of looking for the team's problems, we tried to understand their expectations from specific practices or agile methods before the adoption. This allows us to answer RQ2.2;
- **Benefit**: Again the same process was followed. Instead of looking for the team's problems, we tried to understand the team's benefits after the adoption. This allows us to answer RQ2.3.

5.4 Results

As seen in Figure 5.3, we found that more than 60% of the selected papers were published in the field of agile methods and highly ranked conferences (A- or B-based on Core Portal Conference — http://portal.core.edu.au/conf-ranks/) including ICSE, HICSS, XP, AGILE, and PROFES. In addition, most of these papers were published less than 10 years ago. We also noticed that more than

70% of the studies in the dataset were conducted in IT companies while the rest were in the IT sector of a non-IT company. Furthermore, it is noticeable that agile methods are tailored and used mainly in a normal agile environment (51%), distributed environment (23%), and Scaled Agile (8%). As a result, we can conclude that our dataset is representative for our study.

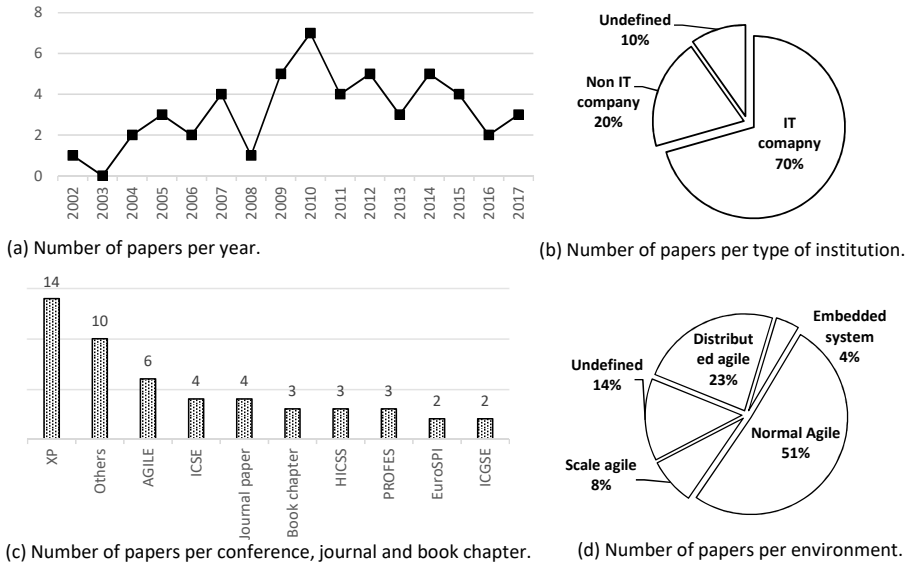


Fig. 5.3 Dataset information.

5.4.1 RQ1: How have the Agile Manifesto and its influence been discussed in tailored agile methods adoption?

Figure 5.4 summarizes the result of our analysis from the 51 papers.

Overall, 51% (26 papers) of the papers mention the Agile Manifesto when evoking the tailored agile methods adoption (see Figure 5.4.a). Furthermore, all the papers do not discuss the Agile Manifesto in the same way. 38% of them (10 papers out of 26) simply mention the word "Agile Manifesto" without even referring to neither a value nor a principle. The other 38% refer to only the values, 12% (3 papers) refer to only the principles and the rest 12% refer to both values and principles (see Figure 5.4.b).

Concerning the influence of the Agile Manifesto on tailored agile methods adoption (see Figure 5.4.c), it is only discussed in 14 papers among which 11 papers (42%) acknowledge it with a clear explanation while the other 3 (12%) only acknowledge without further details.

The result of RQ1 shows that the interest of the Development Team in understanding the Agile Manifesto is not significant. Overall, out of 51 case studies, 51% talk about it while only 21% (11 papers) acknowledge its influence.

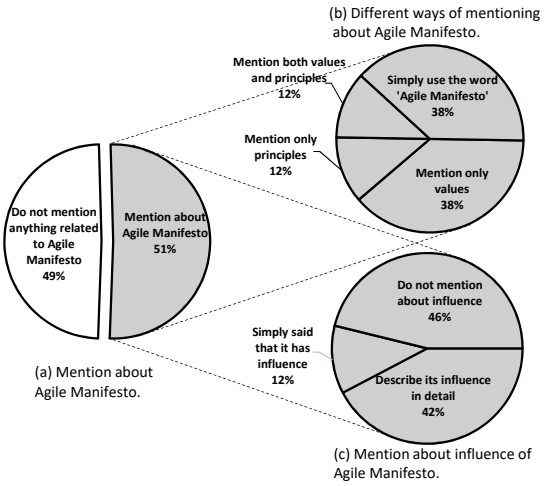


Fig. 5.4 The influence of the Agile Manifesto in tailored agile methods adoption.

5.4.2 RQ2: Is the Agile Manifesto related to agile practices selection?

To answer RQ2, our intuition was to compare the problems, expectations, and benefits extracted from the 51 selected papers with the Agile Manifesto, i.e., the 4 values and 12 principles. Since we had already provided the mapping among the 4 values and 12 principles in [86] (see Table 5.2), we only compared them (problems, expectations, and benefits) with the 12 principles.

From the data extraction process, we gathered 3 lists of statements, one for the problems[1], one for the expectations[2], and one for the benefits[3]. As a result, we have 42 statements describing problems, 155 statements describing expected results, and 205 statements describing benefits.

The mapping process was carried out manually by one author and double-checked by another, in the form of a Cartesian product. This means that for each list, we compared every statement to the 12 principles of the Agile Manifesto. They are mapped when they have a close relation to one another. For instance, the problem "delivery pains" is closely related to both Principle_6 "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software" and Principle_7 "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale". This problem is thus mapped to both the Principle_6 and the Principle_7. The result of the mapping is exposed in Figure 5.5.a. Figure 5.5.b provides the mapping of problems, expectations, and benefits to the 4 values. The number of problems, expectations, and benefits, which were mapped to the values, is the result of the union between the different principles contributing to each value.

[1]Problems were extracted from 12 papers that described the problems they encountered which led them to tailored agile adoption.

[2]Expectations were extracted from 27 papers that discussed the team's expectations.

[3]Benefits were extracted from 37 papers that discussed the benefits of tailored agile methods adoption.

Table 5.2 Mapping agile values and principles.

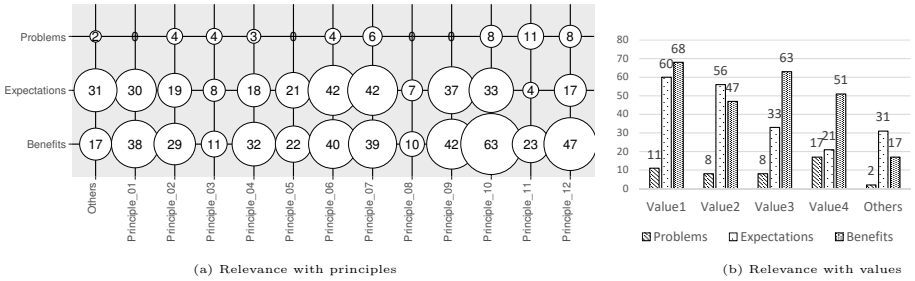
Value	Principle
Value1: Individuals and interactions over processes and tools	**Principle_5**: Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. **Principle_6**: The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. **Principle_8**: Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. **Principle_11**: The best architectures, requirements, and designs emerge from self-organizing teams. **Principle_12**: At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
Value2: Working software over comprehensive documentation	**Principle_1**: Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. **Principle_3**: Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. **Principle_7**: Working software is the primary measure of progress. **Principle_10**: Simplicity--the art of maximizing the amount of work not done--is essential.
Value3: Customer collaboration over contract negotiation	**Principle_4**: Business people and developers must work together daily throughout the project.
Value4: Responding to change over following a plan	**Principle_2**: Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. **Principle_9**: Continuous attention to technical excellence and good design enhances agility.

The final lists of problems, expectations, benefits, and the result of mapping with Agile Manifesto are available online at https://goo.gl/rrghEH.

The correlation ratio between the Agile Manifesto and agile methods adoption goal (problems, expectations, and benefits) is defined by the number of problems, expectations, and benefits that can be mapped to at least one agile principle over the total number we found. For instance, 40 out of 42 problems can be mapped to at least one of the principles. The correlation ratio between problems and the Agile Manifesto is thus 95%.

5.4.2.1 RQ2.1: Is the Agile Manifesto relevant to the team's problems that led to tailored agile methods adoption?

As seen in Figure 5.5.a, most of the problems (95%) can be mapped to the 12 principles. The top line in Figure 5.5.a shows the distribution of problems in



(a) Relevance with principles (b) Relevance with values

Fig. 5.5 Mapping of problems, expectations and benefits with Agile Manifesto.

line with related principles. The three most relevant ones are Principle_10, Principle_11, and Principle_12. The reason is that most problems faced by the Development Team are customer-based and the change-oriented ones that motivate tailored agile methods adoption. Four principles are not mapped with any problem. However, at the value level, we can see in Figure 5.5.b that all the values are relevant.

We can summarize that the Agile Manifesto and the team's problems are closely related to one another. However, the number of problems is not significant; it leads us to conclude that problems faced by the Development Team are not the main reason for tailoring agile methods for adoption.

5.4.2.2 RQ2.2: Is the Agile Manifesto relevant to the team's expectations from tailored agile methods adoption?

We extracted 155 expectations in total from the selected papers. Figure 5.5.a (second row) and Figure 5.5.b present the detailed statistics of agile principles and values respectively. The majority of the expectations (80%) can be mapped to at least one principle. The three most relevant principles are Principle_6, Principle_7, and Principle_9 which contribute to Value2 "working software over comprehensive documentation" i.e., having a working software is always what people expect the most. More precisely, Principle_6 and Principle_7 describe a very similar idea on software delivery and thus they both have slightly different numbers of "expectations". At the value level, Value1 "Individual and interaction over process and tool" is the most relevant. In contrast to the "problems" section, Principle_11 and Value4 are the least relevant. While the differences between the two most relevant values (Value1 and Value2) are not significant, a big gap exists between the most and the least relevant values (Value1 has 60 related expectations and Value4 just 21).

Briefly speaking, we can conclude that the Agile Manifesto is relevant to the team's expectations when tailoring agile methods for adoption. However, agile principles are not all equally important. This corresponds exactly to the motivation of tailored agile methods adoption, i.e., adopting only the most relevant principles or practices instead of full adoption.

5.4.2.3 RQ2.3: Is the Agile Manifesto relevant to the team's benefits of tailored agile methods adoption?

The mapping results between the benefits extracted from the papers and the elements of Agile Manifesto are presented in Figure 5.5.a for the principle level and Figure 5.5.b for the value level. We found that the majority of the benefits (92%) could be mapped to at least one principle. At the value level, Value3 and Value1 are the most relevant among all. This proves that agile methods allow Development Teams to improve their communication both between team members and to customers. Globally, the number of "benefits" mapped to each value does not change much from one value to another. Also, it is noticeable that there is a strong correlation between expectations and benefits.

The overall results show that 95% of problems, 80% of expectations, and 92% of benefits can be mapped to principles and values of the Agile Manifesto. It means that the Agile Manifesto is highly related to the real development of the team's goals in every situation: problems, expectations, and benefits.

5.5 Threats to Validity

Kitchenham [80] states that the systematic process involved in SLR is designed to avoid bias. Thus, in every step of our SLR process, we manage the bias as much as we can.

Starting from the data source, Kitchenham et al. [81] claims that researchers should collect from at least 4 different sources. Inspired by this idea, we collected our data from four different sources: IEEEXplore, ACM Digital Library, SpringerLink, and ScienceDirect. For the keywords used in search engines, we used multiples terminologies (synonyms) used by both researchers and practitioners. We only consider papers published between 2000 and 2017, since the Agile Manifesto could not be mentioned before 2000. Therefore, we unavoidably missed some papers. However, we believe that we have retrieved a large and representative sample for this review. Regarding the inclusion and exclusion criteria, we defined and refined them several times before starting the real selection to collect the maximum relevant papers and effectively reject irrelevant ones. According to Kitchenham [80], this can greatly minimize the possibility of bias. To address the problem of quality, in accordance with Campanelli and Parreiras [32], we only considered peer-reviewed papers from conferences and journals. Next, in the data extraction and classification stages, we applied standard classifications defined in the current literature based on shared and common definitions. We had multiple face-to-face discussions when there were misunderstandings in some concepts.

To summarize, we have considered the internal validity of this chapter to be acceptable. Most of the biases encountered are inherent and we aimed to manage them as much as we could.

5.6 Limitations

There are some limitations in this chapter caused by the biases in the SLR process that we could not avoid.

First, there is no explicit definition of the term "quality" in our article selection criteria. Instead, we simply assume that all conference papers/journals reach an acceptable level of quality.

Second, there is bias in the data extraction where we tried to capture the importance of the Agile Manifesto. Its importance was not always explicitly mentioned but interpreted by the readers. Consequently, some data may have been missed or misunderstood.

Finally, there is bias in the publication trends. Based on Kitchenham and Charters [82], they refer to the problem that positive results are more likely to be published than negative ones. In other words, very few case studies have reported their failures in agile methods adoption. Instead, they have been focusing more on the benefits of the adoption. There are thus many unsuccessful cases we have missed.

5.7 Discussion and Conclusion

The primary aim of this chapter was to verify the relation between the Agile Manifesto and agile practices selection through an SLR approach. We first tried to find out *how the Agile Manifesto has been discussed in tailored agile methods adoption*. Then, we tried to see *whether or not agile practices selection can be related to agile values or principles defined in the Agile Manifesto* by comparing them with the team's problems, expectations, and benefits.

The result of RQ1 shows that our observation is true, the Agile Manifesto has really lost attention from the Development Team. Among the 51 selected papers, only about half of them (51%) mentioned the Agile Manifesto (detail and not detail). Agile practitioners tend to follow only the rules of a specific methodology such as Scrum, XP, etc., and completely ignore the manifesto.

On the contrary, the results of RQ2.1, RQ2.2, and RQ2.3 show that the 4 values and 12 principles of the Agile Manifesto are highly relevant to the team's problems, expectations, and benefits. 95% of problems, 80% of expectations, and 92% of benefits can be mapped to principles and values of the Agile Manifesto. It means that the Agile Manifesto still covers fundamental aspects of any agile method. Therefore, Development Teams should spend some time to understand the Agile Manifesto before adopting any agile method including a tailored one. In addition, as can be seen in Figure 5.5, there is a strong correlation between expectations and benefits (except for the principles Principle_10 and Principle_12). This high correlation can explain that, by tailoring agile methods to meet their expectations, the team can of course obtain the benefits accordingly.

In conclusion, even though a lot of research supports the idea that the Agile Manifesto (values and principles) allows defining the set of practices, yet software Development Teams tend to neglect the Agile Manifesto when tailoring agile methods for adoption. We found however that the Agile Manifesto should be more valued and draw more attention from the Development Team; it deserves

to be a guideline for the Development Team to tailor any agile method and select the right features for adoption. Having a deep knowledge of the Agile Manifesto gives advantages for better tailoring agile methods to maximize the team's expectations and eventually the benefits.

Finally, this study provides a more insightful validation on the relation between the Agile Manifesto and agile practices which was always made based on the assumptions or beliefs of the researchers. This validation can be used as the evidence to create a more complete framework for tailored agile methods adoption in an alternative perspective. For the next step, we aim at building a repository through a systematic review of the empirical studies to gather the relationships between the Agile Manifesto and each practice. Using this repository, the practitioner can then identify easily the related practices to fulfill fully or partially the principles and values of the Agile Manifesto.

Acknowledgments

The authors would like to thank Benjamin Croix for his involvement in the Systematic Literature Review, i.e., the papers selection process, and data extraction.

Chapter 8

Towards a Systematic Socio-Intentional Framework for Agile Methods Tailoring

In this chapter, we propose a framework to analyze agile practice before the adoption, with the help of the OBAMA-tool and modeling technique. We start by explaining the research context, related works, motivation, and the contributions of this chapter in Section 8.1. Each contribution is then described in detail in the following sections. Section 8.2 discusses the first contribution which is about finding the right socio-intentional representation to depict the information on agile methods adoption. The notions and modeling techniques are also highlighted. Section 8.3 describes the second contribution which is the proposed methodology for tailoring agile methods. In every step, we explain how to find relevant information from the tool, how to build the model, and how to analyze agile practice. In the last contribution, we use a case of a real software development project as an example to demonstrate how our framework is applied in practice. This feasibility study is described in Section 8.4. Finally, we provide a conclusion and discuss the future works in Section 8.5.

A preliminary study of our framework for agile methods tailoring is published in *Software Technologies - 12th International Joint Conference (ICSOFT - 2017) proceeding* [83]. Based on the contributions in **Chapter 5**, **Chapter 6**, and **Chapter 7**, we present in this chapter a more sophisticated version of the framework. The following content is based on our article, accepted by the *IEEE International Conference on Business Informatics (IEEE CBI - 2021)*.

8.1 Introduction

After years of experience with agile methods, practitioners find a growing interest towards agile methods tailoring in order to adopt only fragments of different methods based on different criteria. Let us imagine a scenario where a development team wants to adopt agile methods, the right way should be adopting only agile practice that fits their needs and context. Once agile practice is identified, to get the expected result from the adoption, the development team needs to have a strategy to adopt it successfully. In order to facilitate these processes, many approaches have been proposed by focusing on different aspects of the software development such as process, resource, and goal [11, 43,

59, 71, 105, 133]. However, none of them focuses on socio-intentional aspects, precisely the way team members work together to successfully adopt the agile practice and to achieve their goals.

Prior studies that modeled an agile context using socio-intentional aspects in the most convincing way are [56, 59, 11]. Esfahani et al. [56] proposed the usage of the i* modeling framework [54] to visualize the social perspective in agile methods. Even though this work provides evidence to show that the representation of the relationships and dependencies between the role helps analyze agile methods and identifying possible vulnerabilities, it does not provide a clear methodology of how to build the model and how to use them effectively for agile methods tailoring. Later on, the authors proposed a framework for the evaluation of a candidate agile method prior to its enactment within the development environment of an organization [59]. This framework includes the steps to analyze agile practices, identify problems of process and find solutions based on the concept of strategic actors of organizations, and their dependency relations. Their framework includes the steps to analyze whether an agile practice is suitable for the adoption based on the team's objective and situations. Similarly, Ambler and Mark [11] proposed a Disciplined Agile (DA) toolkit that takes a goal-driven approach to guide through the process-related decisions, to tailor and scale agile strategies based on the situation a team faces and the outcomes it desires. These last two frameworks however focus only on the goal aspect. On the other hand, the social aspect is known as being one of the foundations of agile methods. Two main agile values and four main agile principles focus on people, their interactions and collaborations. According to Eckstein [50], most projects do not fail due to technology, but because of social and organizational problems, and a lack of effective communication. In other words, neglecting the dependencies between roles hampers the chance to successfully adopt an agile practice and may cause the team to miss its goal.

Three key problems were identified from existing frameworks:

1. The 4 values and 12 principles of the Agile Manifesto have never been used in any of the frameworks. In **Chapter 5**, we prove that understanding the Agile Manifesto is an effective way to define the outcome of different practices. Nevertheless, the existing frameworks would rather be oriented to business goals, various values, or principles than the ones defined by the Agile Manifesto;

2. Theses frameworks focus on what to do (process), what to produce (product) and what to achieve (goal), but they have hardly discussed any concerns related to the social aspects. For instance, dependencies between team members to conduct a practice, the vulnerabilities caused by the roles, and how they solve the problems, etc;

3. In most frameworks, the tailoring processes are made based on theory or the propositions of the authors. The actual experiences of agile practice adoption, which can be found in the literature, have barely been used. As a consequence, the outcome of real-life practice adoption is different from the expectation.

To address these problems, we propose a socio-intentional framework that focuses on the intentional (why?) and social (who?) dimensions. In this framework, we propose a methodology to analyze agile practices and to define the right strategies for adopting them based on the team's goals, their situations, and dependencies between team members. First, we analyze the relationship between the adoption goals, agile practice, and the suitability of the team for the adoption. In the goal dimension, the Agile Manifesto is used as one of the criteria for agile practice selection. By balancing between the best agile practice that can achieve the desired goals and the most suitable one with respect to the team's situation, we can decide what practice to adopt. Once the right agile practice is identified, we analyze the social requirements (roles, tasks, and dependencies) for adopting it. This step allows us to foresee the vulnerabilities that can lead to adoption failure. Based on that, we can prepare for a successful adoption by solving the vulnerabilities. In our framework, rather than following agile practice adoption prescribed in a theory, we use our evidence-based tool to provide the needed information. This tool was created based on real agile adoption experience found in the literature. By doing so, we bring the results of agile methods tailoring closer to reality. As it is hard to analyze the information listed by the tool in textual format, the information is also presented through a social-intentional modeling language; this is intended to ease the analyzing process.

This chapter is the continuation to the Chapter 5, Chapter 6, and Chapter 7. The contributions in this chapter are divided into three parts as follows:

1. **Socio-intentional Modeling Framework Usage:** to ease the analyzing process, we propose using models to visualize the information provided by the tool. We find suitable modeling language and notions by mapping the concepts and relationships in our ontology model with the notions of different modeling languages that can represent either goal or social dimension. For the agile concepts that cannot be mapped, we propose extended notions for the representation;

2. **Methodology for practices selection and adoption:** to guide practitioners on how to analyze agile practices and prepare for the successful adoption, we propose a well-defined set of steps, where each of them also includes how to use the tool and modeling technique to ease the analyzing process;

3. **Feasibility Study**: to demonstrate how to use our framework, we provide an illustrative example using a case of a real software development project.

8.2 Socio-intentional Modeling Framework Usage

There are several well-known modeling frameworks which have been proposed to visualize goal and/or social aspect, including NFR [37], KAOS [44], i* [165], Tropos [37] and iStar 2.0 [42]. To find the most suitable representations, we performed the Cartesian product of the elements (concepts and relationships) in our ontology model and the notions of different modeling frameworks that can

represent either goal or social dimension. We basically compared every element we wanted to represent with every notion in every selected modeling framework. The best match was the one that had the closest objective/definition. In the comparison, we decide to exclude Tropos as all their notions are either included in the i* or iStar 2.0 framework.

Table 8.1 shows the mapping results between the agile elements (concepts and relationships) and the notions of different modeling frameworks. Based on these results, iStar 2.0 was the most suitable framework as our agile elements match best to its notions.

Table 8.1 Mapping agile concepts and relationships with iStar 2.0

Agile elements	**KAOS**	**NFR**	**i***	**iStar 2.0**
Value	Goal	Soft-goal	Soft-goal	Quality
Principle	Goal	Soft-goal	Soft-goal	Quality
Goal	Goal	Soft-goal	Soft-goal	Quality
Requisite	Requirement	Soft-goal	Soft-goal	Quality
Practice	Operation	Operationalizations	Task	Task
Activity	Operation	Operationalizations	Task	Task
Solution	Operation	Operationalizations	Task	Task
Role	Agent	N/A	Role	Role
Artifact	Entity	N/A	Resource	Resource
Problem	N/A	N/A	N/A	N/A
Situation	N/A	N/A	N/A	N/A
Contribute to	Refinement	Contribution	Contribution (Some)	Contribution (help)
Achieve	Refinement	Refinement	Contribution (Some)	Contribution (Help)
Help	Refinement	contribution (Help)	Contribution (Some)	Contribution (help)
Harm	N/A	Contribution (Hurt)	Contribution (Hurt)	Contribution (Hurt)
Cause	Cause	Contribution (Make)	Contribution (Make)	Contribution (Make)
Perform	Perform	Operationalize	Social dependency	Social dependency
Is responsible for	Responsibility	N/A	Social dependency	Social dependency
Requires	N/A	N/A	N/A	NeededBy
Solve	N/A	N/A	N/A	N/A
Total Mapped Elements	**15/21**	**13/21**	**16/21**	**17/21**

8.2.1 Notion Definitions

For most of the elements, we follow the exact definitions defined by iStar 2.0. We however slightly change the definitions of the relationships *NeededBy* and *Contribution link (Make)* adapting to our needs.

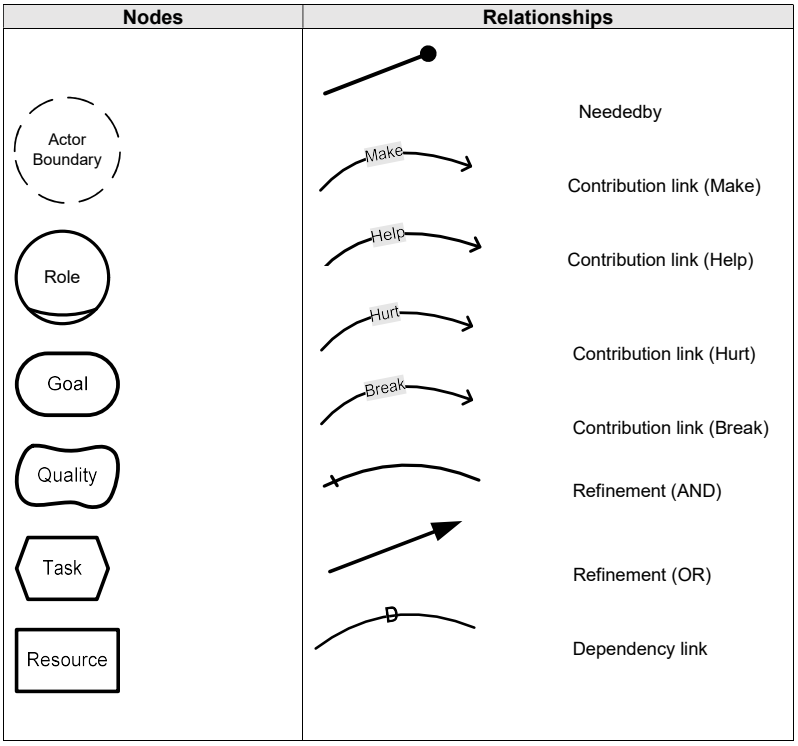


Fig. 8.1 iStar 2.0 notions

Figure 8.1 visualizes the notions of the iStar 2.0. We summarize the definition of the nodes and relationships which are going to be used in our framework hereafter.

8.2.1.1 Nodes

- *Role*: an abstract characterization of the behavior of a social actor within some specialized context or domain of endeavor;
- *Quality*: an attribute for which an actor desires some level of achievement;
- *Task:* an action that an actor wants to be executed, usually with the purpose of achieving some goal;
- *Resource*: a physical or informational entity that the actor requires in order to perform a task;
- *Depender*: an actor that depends for something to be provided;

- *Dependee*: the actor that should provide something.

8.2.1.2 Relationships

- *NeededBy*: a relationship that links goal, task, and resource. It indicates what needs to be accomplished or provided to achieve a goal or execute a task;
- *Contribution links*: a relationship that represents the effects of an element on another. There are four types of contribution link which can be divided into two categories: *Fulfilled* and *Denied*. The first one refers to a sufficient positive evidence (*help*, *make*) while the later one refers to a strong negative evidence (*hurt*, *break*);
- *Refinement links*: a generic relationship that links goals and tasks hierarchically. There are two types of refinement: *AND* and *OR*;
- *Dependency link*: a relationship that represents dependency between *Depender* and *Dependee*.

8.2.1.3 Notion Extension

There are several concepts and relationships in our ontology model which cannot be mapped to any iStar 2.0 notions. We thus extend the representations of the remaining elements as Figures 8.2.

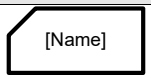
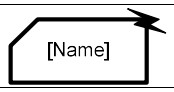
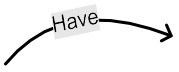
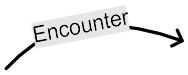
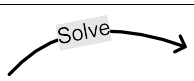
Agile element	Extended Notion	Definition
Situation		A state or situation that can affect (good or bad) another situation, goal or quality.
Problem		A bad state or situation that is conflict with another situation, goal or quality.
Have		A relationship “Have” between role and other elements
Encounter		A relationship “Encounter” between role and other elements
Solve		A relationship “Solve” between solution and problem

Fig. 8.2 Example of *Sprint planning* in SD view

8.2.2 Modeling Technique

Our purpose is to visualize all the information provided by the tool in models. By all means, the visualization should help practitioners analyze agile practices and identify what practices they should adopt and how to adopt them successfully.

iStar 2.0 framework guides us to visualize socio-intentional perspective in multiple model views, that include *Strategic Dependency*, *Strategic Rationale* and *Hybrid*. The first two model views fit very well with our needs. We however need to use another model view inspired by the NFR framework to fit some of our cases.

In the following sections, we explain how these model views can be used in our framework.

8.2.2.1 Strategy Dependency

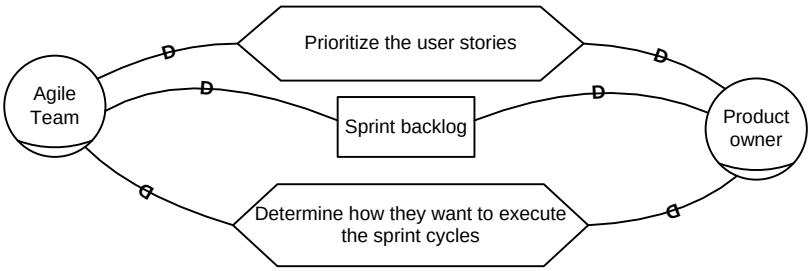


Fig. 8.3 Example of *Sprint planning* in SD view

The SD model describes external dependency relationships between actors, from depender to dependum to dependee, to achieve a goal, quality, task, and resource. This view is suitable to briefly describe the dependencies between roles in the team to achieve the functional and/or the non-functional goal, to perform the activity, and to provide the resource needed during the development. The relationships between elements within each actor are however not visualized in this view. For example, in the practice Spring planning, the Agile team depends on the Product Owner to provide the *Sprint backlog* and *prioritize the user stories* while the Product Owner depends on the Agile team to *determine how they want to execute the sprint cycles*. This example is visualized in SD view as shown in Figure 8.3. This model is used when we focus on the dependencies between dependers and not the details about the relationships between dependums and other elements.

8.2.2.2 Strategic Rationale

The SR view shows all the details captured in the model, including actors, dependencies, actor association links, and the internal details of each actor. Modelers can view the strategic rationale inside each actor in the model. This model is a perfect way to visualize how to conduct an agile practice as it provides a deep representation of the internal, intentional dependency with the refinement and contribution links to describe the stakeholders' interests and the (combination of) activities, sub-goals, artifacts that help achieve the main goal.

For example, to conduct a practice *Daily meeting*, Development Team depends on Scrum Master to have an *effective meeting* while Scrum Master depends on Development Team to build an *self-motivated team*. To conduct an

effective meeting, Scrum Master has to *arrange the meeting every morning* and *gather everyone in the team*. To build a *self-organizing team*, Development Team has to *stand-up voluntarily to report their problems* and *suggest the solutions for any posted problems* by writing on *white board*. This example can be visualized in SR view as shown in Figure 8.4. Using this model, we can see both the dependencies between dependers and the detail about the relationships between other elements.

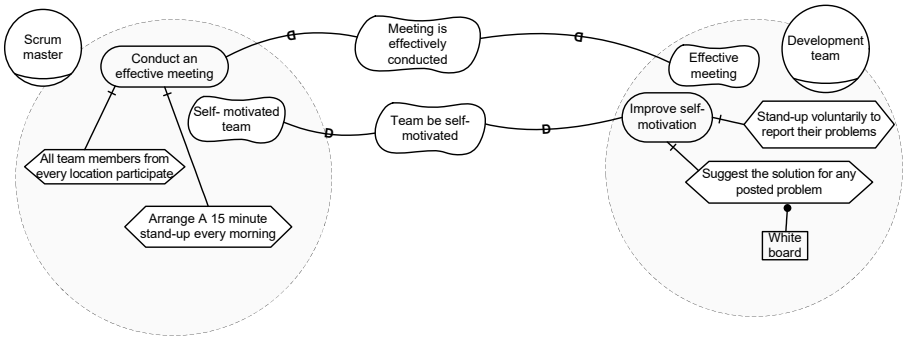


Fig. 8.4 Example of *Daily meeting* in SR view

8.2.2.3 NFR Framework

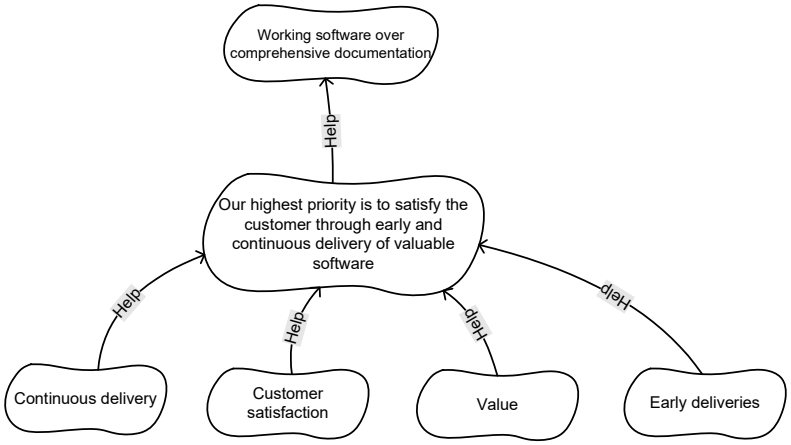


Fig. 8.5 Example of the relationship between agile value, principle, and goal in NFR Framework

iStar 2.0 model views allow us to visualize stakeholders' goals, the combination of activities, sub-goals, artifacts that help achieve the goals, etc. One minor problem of these views is the obligation to include actors in the model, which is not applicable in some steps of our framework. For instance, in the *defining goal* step, the purpose is to visualize dependencies between goals and agile practices.

No actor should thus be included. Inspired by the NFR framework, we propose to represent the relationships between iStar 2.0 notions and the ones we propose without involving any actor. Figure 8.5 is an example of how we visualize the relationship between agile values, agile principles, and goals.

8.3 Methodology for Tailoring Agile Methods Adoption

Tailoring agile methods adoption aims at choosing the right practices to achieve one's goals and integrating them successfully into the software development process. Our socio-intentional framework is divided into two different levels: *Tactical* and *Operational*. The *Tactical Level* helps the team decide what agile practice they should adopt based on what they want to achieve and the suitability of the selected practices to the team's situations. The *Operational Level* helps the team prepare for the successful adoption by identifying beforehand the vulnerabilities in the agile practice adoption and suggesting how to avoid or solve them.

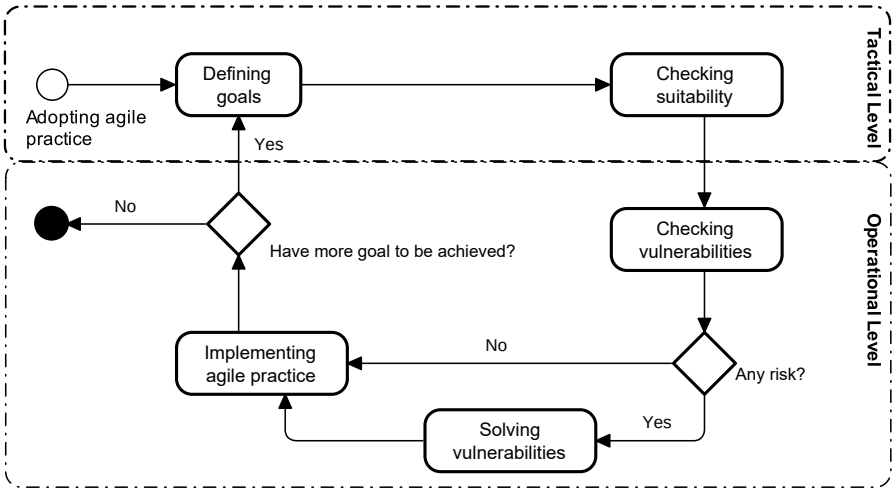


Fig. 8.6 Goal-oriented tailoring agile adoption process.

Figure 8.6 visualizes the whole process of tailoring agile methods for adoption using our framework. The description of each step is depicted hereafter.

Figure 8.6 illustrates the whole process of tailoring agile methods adoption using our framework. The description of each step is depicted hereafter.

- **Step 1: Defining Goals**. First, the software development teams needs to define what they want to achieve by adopting agile practices. It can be something they want to achieve for a better software development process and/or they want to have the agile mindset as defined in the agile manifesto (agile values and principles). Based on the defined objectives, they can then identify the suitable practices;

- **Step 2: Checking Suitability**. Team's situations can affect the result of agile practice adoption. To identify whether a team is suitable to adopt a practice, practitioners need to define situations of the team and analyze the impacts (*Help* or *Hurt*) on the agile practice adoption;
- **Step 3: Checking Vulnerabilities**. To successfully integrate the agile practice into their development process, the team members need to perform their activities correctly and avoid the vulnerability as much as possible. In our framework, vulnerabilities can be identified by analyzing (1) various dependencies between roles to perform activities and (2) problems encountered in previous experiences. The vulnerability happens when (1) any role fails to do its activities, (2) the artifact required is not sufficiently provided, and (3) when problems identified in the literature might happen in their team;
- **Step 4: Solving Vulnerabilities.** Based on the results from step 3, if any vulnerability is identified, the team should avoid it by reinforcing the roles in charge, provide the required artifacts and apply the solution to the expected problems;
- **Step 5: Implementing Practices.** The team can implement the practices into their development process by ensuring that the activities which are parts of the practice and the tasks to avoid the vulnerabilities are accomplished.

In every step of the framework, practitioners start by finding the relevant information from the OBAMA-tool. This information is then illustrated in iStar 2.0 before it can be analyzed. In most steps, practitioners can build the model by simply representing the relevant information with the corresponding iStar 2.0 nodes or links, as mapped in Table 8.1 or the extended notions as shown Figure 8.2. For instance, in step 1, the information about *goal*, *agile value* and *agile principle* are represented by the *quality* node; and *practice* is represented by the *task* node. The relationships between nodes are represented by the *contribution link Help*. Except for step 3, practitioners need to analyze the purpose of each activity before they can build the dependencies between actors.

In the following sections, we explain in detail each step of the framework.

8.3.1 Defining Goals

In the first step of our framework, agile practitioners need to define what they want to achieve by adopting agile practices. The reason behind their adoption can be some random goals related to software development and/or the agile mindsets (Agile Manifesto).

Goal can be something a team intuitively wants to achieve. It can also be derived from the problems or the obstacles within their current development process which they want to overcome. For instance, if a team has difficulty in understanding the requirement because the customer is hardly available for the discussion, then we can derive these problems into multiple goals such

as *improve understanding customer's needs* and *improve communication with customer*, etc. Besides *goal*, there is a set of the agile mindset defined in the Agile Manifesto, which can also be achieved by agile practices. The relationships between the Agile Manifesto (*value*, *principle*) and agile practice are explained in Chapter 6 and Chapter 7.

To find the information related to what a practice can achieve, we choose three following concerns: (1) *the goal team can achieve by adopting a practice*, (2) *the agile value a team can achieve by adoption a practice*, and (3) *the agile principle a team can achieve by adopting a practice*.

To build the model based on the information provided by the tool, we represent the information about *goal*, *agile value* and *agile principle* by *Quality* node, and *practice* by the *Task* node. We then build the relationships between nodes (practice and goal, goal and principle, principle and value) by using the *Contribution link*.

To analyze what practice we should adopt, we need to check how many goals are contributed by each practice. It also depends on how we prioritize the goals we want to achieve. A suitable practice should be the one that contributes to most of the top-prioritized goals.

An illustrative example of how to get the information, build the model, and analyze agile practice at this step can be found in Section 8.4.1.

8.3.2 Checking Suitability

This step helps practitioners decide whether or not they should adopt specific practices based on team's situations. Each agile practice has a list of the pre-conditions (also known as *Requisite*) required in order to adopt it successfully. These requisites are impacted by the team's situations [58]. To know whether a team is suitable to adopt a practice, we thus check the suitability based on its situations.

To find the relevant information about the impact of the team's situations on practice, we follow two following steps:

1. Describe team's situations and select the practice(s) team wants to adopt in the *Input page 2*. The situational factors listed in this page are agile practice selection criteria defined in [32], on the basis of an SLR;

2. In the page *Information related to practice based on input*, we choose the concerns (1) *the situation of the team which is bad for adopting a practice* and (2) *the situation of the team which is good for adopting a practice* to know how team's situations impact each practice.

To build a model that allows analyzing the suitability of a practice, we represent the information about *practice* by *Task* node, *requisite* by *Quality* node and *situation* by *Situation* node. We then connect *practice* and *requisite* by using *NeededBy* link, and connect *requisite* and *situation* by using *Contibution link (Hurt or Help)*.

To analyze what practices are suitable for the team, we need to check how *requisites* are *hurt* or *helped* by the situations. We then denote the status of each requisite based on how many situations have good or bad impact to it.

These impacts lead to the accumulation of evidence for a requisite to be satisfied (✓), weakly satisfied (✓̇), weakly denied (✗̇), or denied (✗). The most suitable practice is the one that has all the requisites *satisfied* by the team's situations.

An illustrative example of how to the information, build the model, and analyze agile practice at this step can be found in Section 8.4.2.

8.3.3 Checking Vulnerability

To check the vulnerability, we need to know the activities we need to perform as well as the relationships, particularly dependencies between roles along with the artifacts required. At the same time, we also need to know the possible problems they may encounter so that they can find alternatives to avoid them.

8.3.3.1 Dependency between roles in performing the activities

Each agile *practice* is composed of many *activities* which require a lot of dependencies, back and forth, between the roles as well as the artifacts. To gain the full benefit from adopting these practices, it is important to perform all the activities correctly and avoid all possible risks or failures. The information and visualization of the dependencies allow us to see what roles and artifacts are critical and then identify the possible chances of failure. For instance, if many activities depend on a particular role and a person who plays that role is not reliable, the probability of failure will increase.

To find the relevant information about activity, role and artifact, in the page *Information related to practice based on input*, we choose the concern (1) *the activity a team should perform as part of a practice*, (2) *the roles or responsibility distribution needed for a practice* and (3) *the artifact required for adopting a practice.*

To build a model that can capture detailed information, including dependencies between actors, actor association links, and the internal details of each actor, we need to use SR model view. We start by representing each *agile role* by *Actors Boundary*, *activity* by *Task* node and *artifact* by *Resource* node. We then put *activities* in the *Boundary* of the *Actor* in charge. After that, we group the *activities* based on the *goals* they contribute to. We then build the relationship between *activities*, *resources* and *goals*. We represent the relationship between *activities* and *goals* by *Refinement* link, and between *activity* and *artifacts* by *NeededBy* link. Finally, we build the *Dependency links* between *roles* based on what *goals* each role wants to achieve and which *role* is in charge.

To analyze the vulnerability of practice, we need to analyze all dependencies, either between roles or resources. There is vulnerability when there is dependum that cannot be satisfied.

An illustrative example of how to the information, build the model and analyze agile practice at this step can be found in Section 8.4.3.1.

8.3.3.2 Problems team may encounter and Solutions

Many case studies on the agile practice adoption experience are usually described with a list of problems the team has encountered, and sometimes with the solutions. We can identify the vulnerability when any of the reported problems will likely happen in the team.

To get the information about causes and problems, we choose the concern *the problem a team may encounter while adopting a practice* on the page *information related to practice based on input*.

To build a model to represent the causes and problems, we use the NFR model view. We start by representing the *cause*, either by *Task* node or *Situation* node, according to the type of the cause. We then represent *problem* by the *Problem* node. Finally, we connect the *cause* and the *problem* with the *Contribution link (Make)*.

In the analysis, we focus more on the problems caused by the team's situation or activities it has to perform. Some situations or activities can cause more than one problem. For the problems which were reported without any defined cause, we need to check whether or not they can probably happen in the team. The more problems will likely happen in the team, the higher chance it fails to adopt the practice. Identifying the causes and the problems allows us to find beforehand the effective ways to avoid them.

An illustrative example of how to the information, build the model and analyze agile practice at this step can be found in Section 8.4.3.2.

8.3.4 Solving Vulnerabilities

Besides the activities which are parts of agile practice, team members need to carry out some extra tasks to avoid the identified vulnerabilities. For instance, to avoid the vulnerabilities that may happen while performing *Daily meeting*, the *Scrum Master* needs to guide other team members on how to correctly perform their task and provide all the necessary resources. To prevent other vulnerabilities found in the literature, practitioners can use solutions provided by our tool or/and figure out new ones.

To find the relevant information about the solutions and the role in charge, we choose the concern *the solution a team may use to solve the problem*.

To build a model to represent the solutions to the problems, we simply add the *solutions* to the diagram in the Step 3 by using *Solution* node, and then connect *solution* to the *problem* by using *Solve* link. This diagram allows us to brainstorm and identify the suitable solutions to avoid the vulnerabilities. After identifying all the solutions, we conclude all the necessary activities by building another model in SD view. We start by representing the *role* in charge by the *Role* node, and *activities* by *task* node. We then build the relationship between *roles* and *activities* by using *Dependency link*.

In the analysis, we focus on finding the solutions to solve the problems. A good solution should be effective and applicable to a team's situation. In addition, the distribution of the role in charge of the solutions is also important. We need to assign the right role for the right task so that the vulnerabilities can be avoided.

An illustrative example of how to the information, build the model and analyze agile practice at this step can be found in Section 8.4.4.

8.3.5 Implementing Practice

After all the analysis in previous steps, the team can start implementing the practice into their development process. To successfully adopt a practice, the team needs to improve the situations which identified as harmful for adoption. Team members have to work together to accomplish all the activities as part of a practice. In addition, they have to perform some extra activities to avoid the identified vulnerabilities.

An example of how to use the models from the previous steps during the implementation can be found in Section 8.4.5.

8.4 Feasibility Study

In order to illustrate the theoretical framework, we use the example of a genuine Development Team made of 4 members that has to adopt agile practices for the development of a software project in a small enterprise. This situation is taken from a real-life application in the context of a training program supported by the university of one of the authors. More precisely the Development Team made of master-level students (i.e. interns) had to develop a timetable application by using PHP and MySQL. These interns have no experience with any software development methodology. Since the company has been using agile methods for a long time, they preferred the interns to develop their projects using Scrum. However, the company let the interns flexibly decide on how they could adopt Scrum in their team. To help the interns adopt agile practice successfully, we suggested them to use our framework and tool as presented in this paper. We asked them to describe team's situations and goals. The internship was taken during the Covid-19 pandemic making it impossible for us to proceed in a full validating case study fashion. We however use the case as a relevant illustrative example where enough data was collected and the relevancy allowed us to evaluate the framework applicability. The description of their situations and goals are indeed complex enough to demonstrate how to use the framework.

According to their description, the company provided good management support for the team. As the team was new to agile methods, a training on Scrum was provided by the company. In addition, the company assigned an experienced Scrum Master and a highly available product owner to work with them. The requirements had already been collected at the moment the team was formed and were expected to be stable at the start of the project. Even though these interns were new to agile methods but they had good knowledge in both the work domain and technology. While the interns were assigned to work at the same company site, they had to work remotely and face-to-face communication was not possible due to the pandemic. They thus used a videoconferencing tool provided by the company. It has been challenging for them to work from home and their goals were therefore to have good communication, be self-motivated, be self-organizing and be punctual. At the same time, they hoped to have a high success rate by clearly understanding what the client needs. To achieve these goals, they believed they also needed to have a clear and realistic task planning which was completely transparent to all team members and the client. The agile value they wanted to achieve by adopting agile practices was to make

individuals and interactions more important than processes and tools. For the principle, they wanted to deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale. We present the application of our framework in this context hereafter.

8.4.1 Defining Goals

To find the practices that can achieve their goals, practitioners can get the information from three concerns in our tool: (1) *the team's goal can be achieved by adopting a practice*, (2) *the agile value can be achieved by adopting a practice* and (3) *the agile principle can be achieved by adopting a practice*.

By choosing a concern in our tool, the information will be displayed in a table format as shown in Figure 8.7, where the number of columns varies based on the concern. In this figure, it is a list of goals achieved by adopting agile practices, where the blue rectangles highlight the information related to the case study. The practitioners can also find the practice to achieve the agile values and principle by selecting them in the *Input page 1* as shown Figure 8.8 to get the results as shown in Figure 8.9.

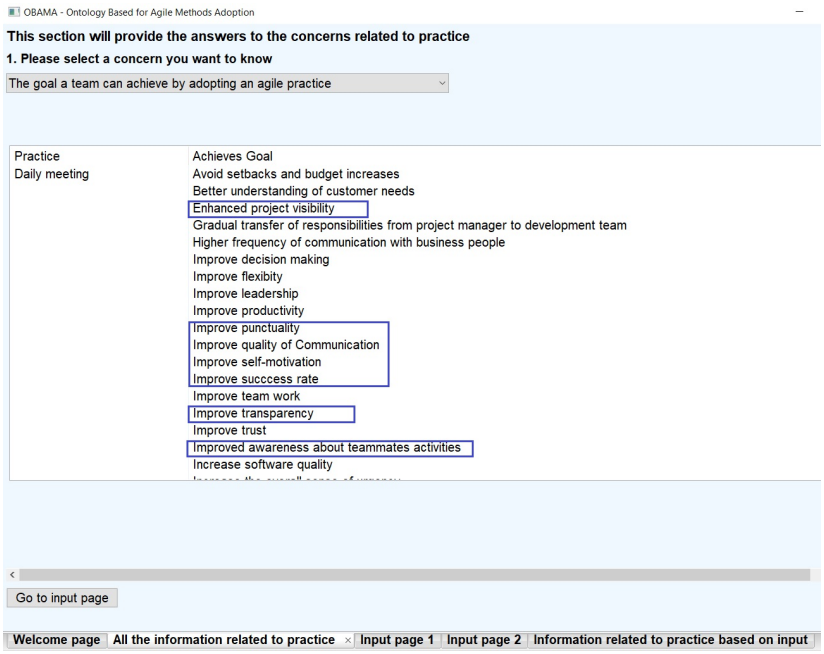


Fig. 8.7 Relationship between goals and agile practices listed by OBAMA-Tool

Based on the results of these three concerns, practitioners can build an iStar 2.0 model as shown in Figure 8.10. In this diagram, we can easily see that three practices contribute to the defined goals and respect the Agile Manifesto. Among these practices, *Daily Meeting* and *Short Iteration* allow achieving the highest number of goals while *spring planning* allows achieving the least. Based on this diagram, practitioners can decide what practice they should adopt

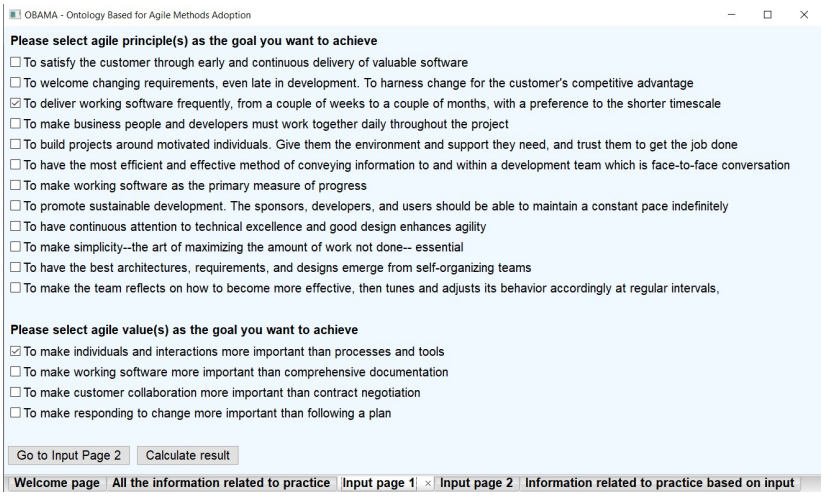


Fig. 8.8 Input page 1 of OBAMA-Tool for agile values and Principle

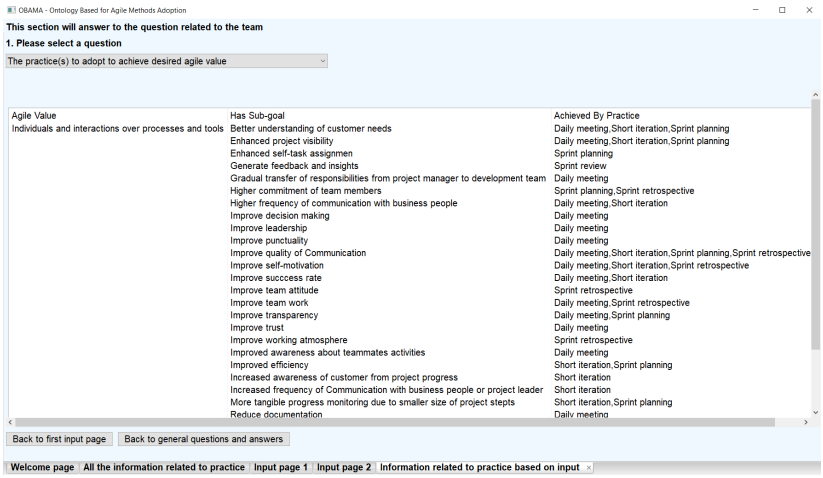


Fig. 8.9 Practices suggested by OBAMA-Tool to achieve selected Value

accordingly. For instance, they can decide to adopt *Daily Meeting* and *Short Iteration* as they allow achieving most goals, or *Sprint Planning* if the goals achieved by this practice are more prioritized, or all together if the situations of the team are suitable.

In the following steps, *Daily meeting* will be used to present our framework. This practice is chosen because it is the most popular practice and it can achieve the most defined goals. The visualizations in iStar 2.0 of another practice, *Short Iteration*, as another example can be found in Appendix D.

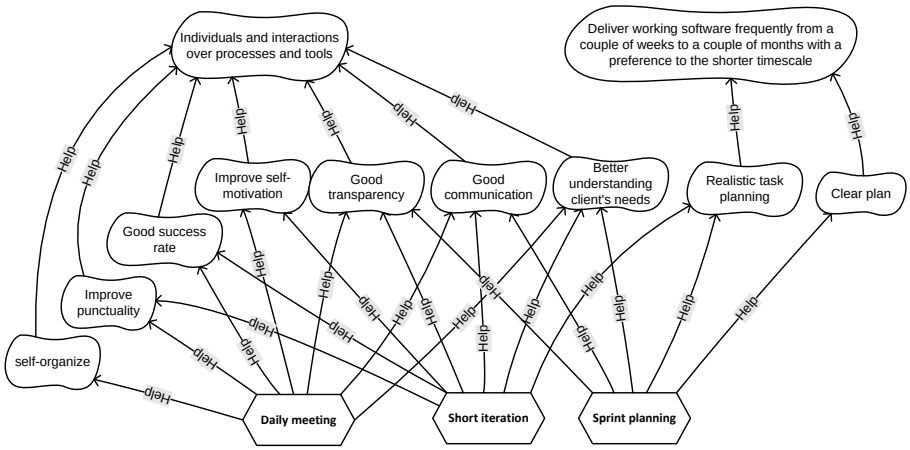


Fig. 8.10 Relationship between agile values, principles, goals and agile practices represented in iStar 2.0

8.4.2 Checking Suitability

To analyze the suitability of the team, practitioners need to describe their situations and choose the relevant practice in "Input Page 2" in our tool. Figure 8.11 is an example of how to describe the above case study, where *Daily Meeting* is the practice the team wanted to adopt. After that, by clicking on the button *Calculate Result*, practitioners can find the relevant information on the page *Information related to practice based on input*. Figure 8.12 is a list of team's situations that have a good impact on *Daily Meeting*, whereas Figure 8.13 is a list of team's situations that have a bad impact.

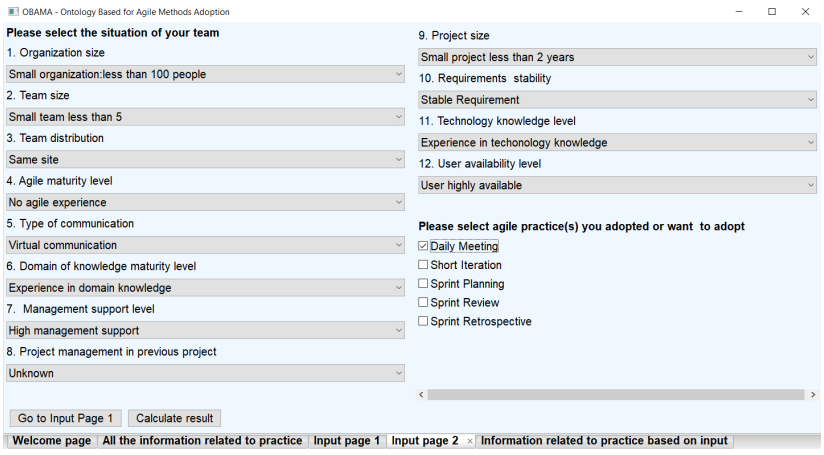


Fig. 8.11 Describing team's situations by using *Input page 2* of OBAMA-Tool

Based on the information provided by the tool about the impact of the team's situations, we can visualize the team's suitability for adopting *Daily*

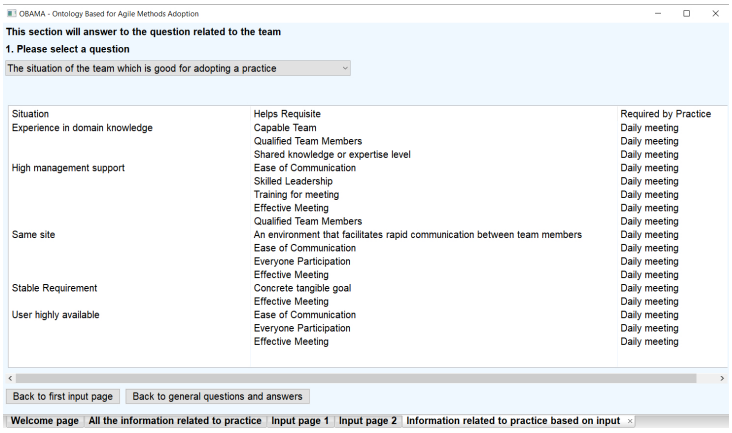


Fig. 8.12 Result of team's situations which is good for practice listed by OBAMA-Tool

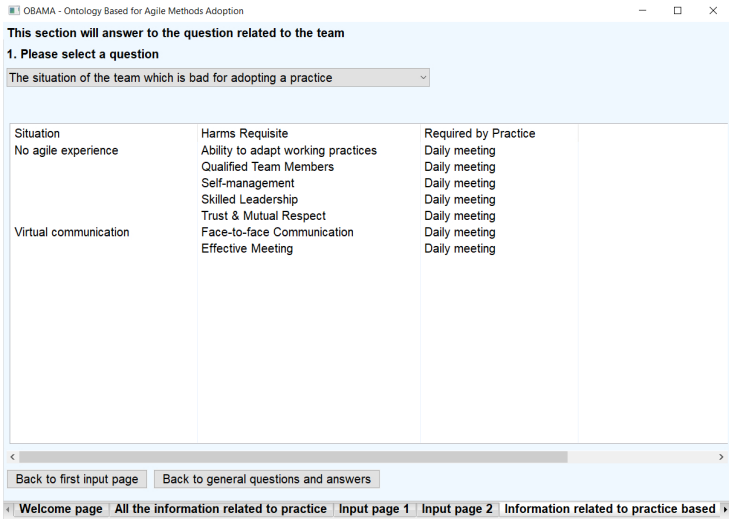


Fig. 8.13 Result of team's situations which is bad for practice listed by OBAMA-Tool

Meeting as shown in Figure 8.14. In this figure, the contribution relation between situations and requisites [(✓), weakly satisfied (✓̇), weakly denied (✗̇), or denied (✗)] are also denoted. Based on Figure 8.14, though *Daily Meeting* is not entirely suitable for the team as two situational factors *Hurt* the requisites, it is still highly possible for the team to adopt it since other situational factors help to *satisfy* and *weakly satisfies* many others requisites. While our tool and diagrams allow practitioners to analyze the suitability of a practice based on their situations, the final decision is on the practitioners themselves. They can still choose to adopt a practice that is not entirely suitable by improving their situations or find alternative practices to achieve their goal.

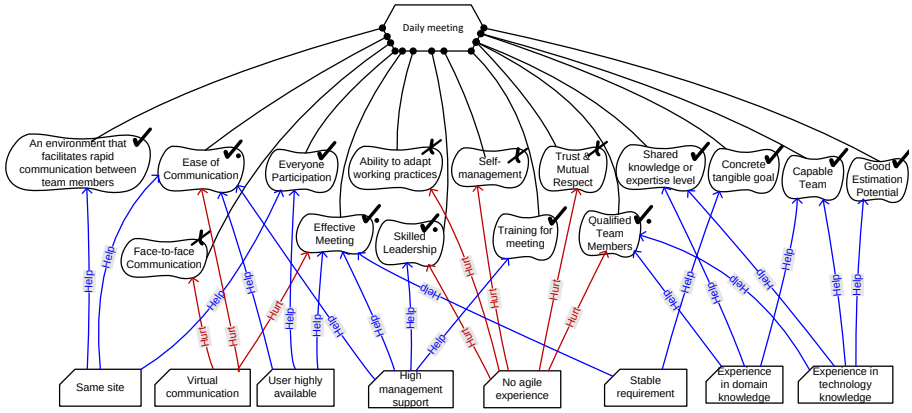


Fig. 8.14 Relationship between Daily meeting, the requisites for its success and team's situation visualized in iStar 2.0

8.4.3 Checking Vulnerability

8.4.3.1 Dependency between roles to perform the activities

It is important for a team to accomplish all the activities and avoid all possible risks or failures. On the page *Information related to practice based on input*, three concerns allow practitioners to find the information about the activity, role, and artifact corresponding to the team's situations: (1) *the activity a team should perform as part of a practice*, (2) *the roles or responsibility distribution needed for a practice* and (3) *the artifact required for adopting a practice*.

Figure 8.16 shows the list of *activities* as part of the *Daily Meeting*, whereas Figure 8.16 shows the list of *roles* in charge. Based on the information provided by our tool, there are four roles required. However, since neither the team in our case has *Project manager* or this role is in charge of many activities, we thus keep only three roles to take charge of the activities.

- Product Owner: the role who clearly knows about the requirement of the software;
- Scrum Master: the role who coordinates the different activities and other members to ensure that development is running smoothly and all the activities are performed correctly;
- Development Team: the group of people who perform a set of activities to successfully build software.

Based on the information within these three concerns, practitioners can visualize the dependencies, back and forth, between roles as shown in Figure 8.17. This diagram allows us to easily see that both the *Scrum Master* and the *Development Team* are responsible for almost the same number of tasks.

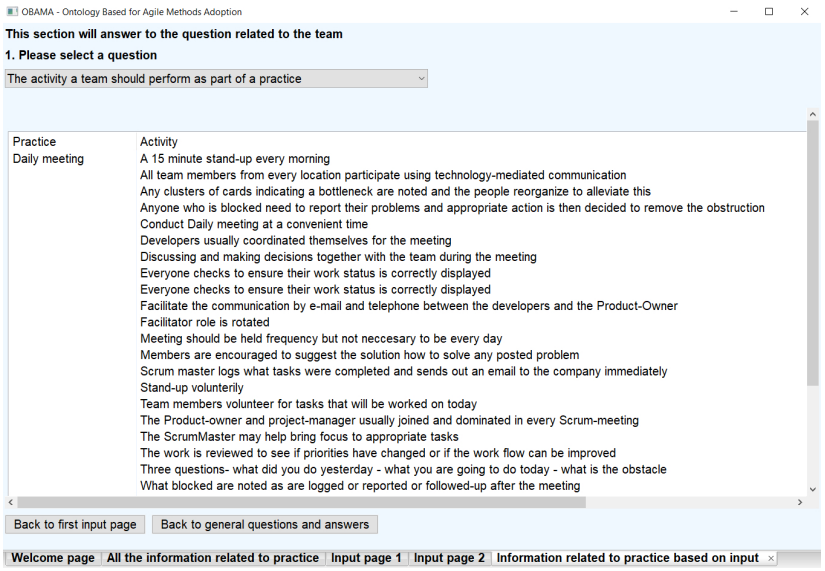


Fig. 8.15 Activities as part of *Daily meeting* listed by OBAMA-Tool

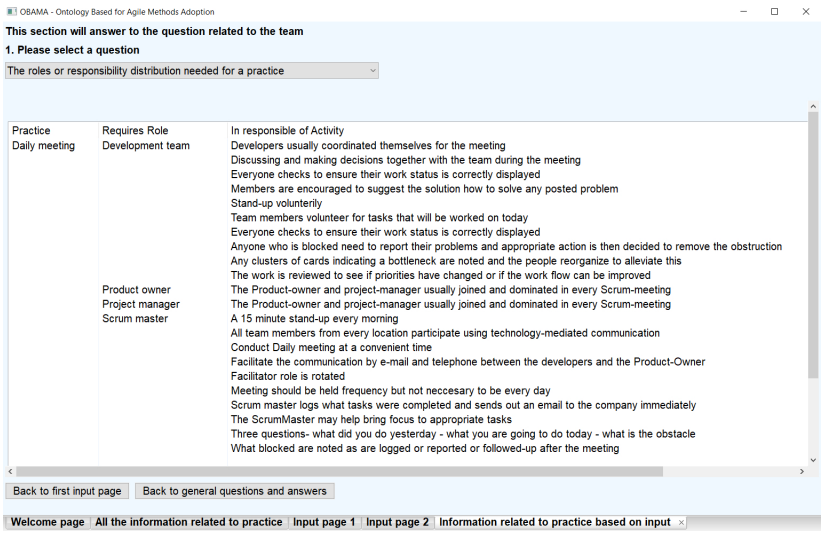


Fig. 8.16 Role required to adopting *Daily meeting* listed by OBAMA-Tool

However, if we go through each task, we can see that the vulnerabilities most likely to happen on the *Development Team* side. One of the reasons is because the *Scrum Master* is experienced with agile which means he already knows how to accomplish his tasks. Besides, the tasks under the *Scrum Master*'s responsibility are practical and easily achievable. On the other hand, the *Development Team* is responsible for the tasks which can be seen as difficult, for instance the task "Stand-up voluntarily" and "Developers usually coordinate

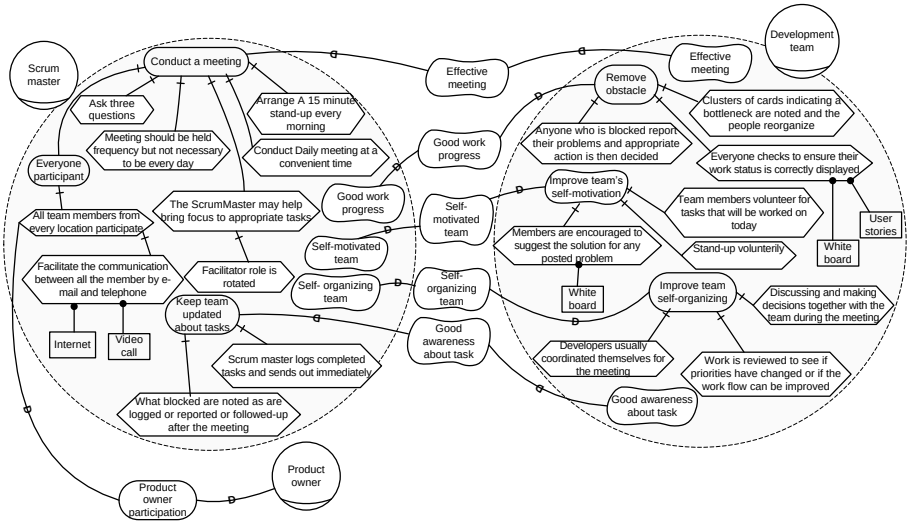


Fig. 8.17 Dependencies between roles to perform activities of *Daily meeting* visualized in iStar 2.0.

themselves for the meeting". These tasks require courage, motivation, and leadership which are difficult to achieve for a new team with no agile experience. These are thus the vulnerabilities that the development team must avoid. It is also important to notice that the resources required such as the "Internet" and "video call" can also be vulnerabilities. For a team whose communication is mostly virtual, failing to provide these resources means failing to conduct a meeting effectively.

8.4.3.2 Problems team may encounter

The practitioners can find the filtered information about the causes and problems by choosing the concern *the problem a team may encounter while adopting a practice* in the page *Information related to practice based on input*. Figure 8.18 list the information about the causes and problems.

The information about the causes and problems can be visualized as shown in Figure 8.19. The relationships between causes and problems are visualized by the new concepts which have been introduced in Section 8.2.1.3. In this figure, many problems have been identified in the previous experiences of adopting *Daily Meeting*, where most of them happened for no specific reason. It can be due to the impossibility to identify the specific reason that causes the problems. For instance "Hold meeting too frequently" and "Board meeting attitude" can always happen when we do not pay attention to how we conduct the meeting. Even though the cause is not identified, these problems are still vulnerabilities that need to be prevented.

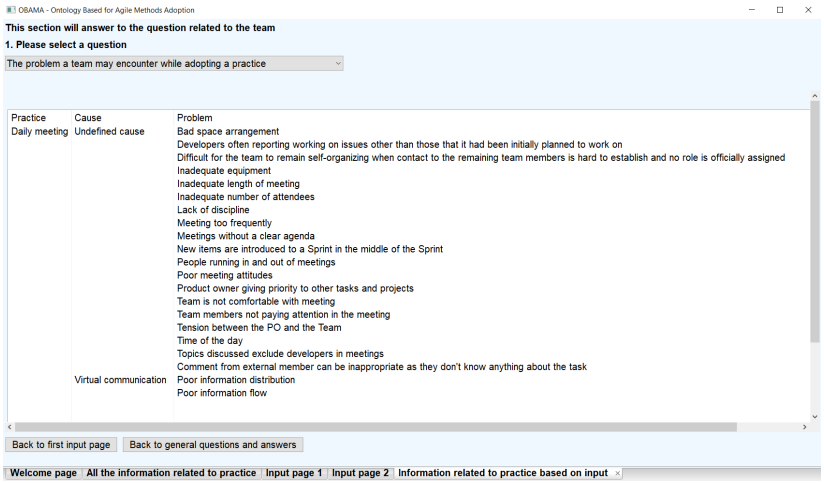


Fig. 8.18 Cause, Problems in *Daily meeting* listed by OBAMA-Tool

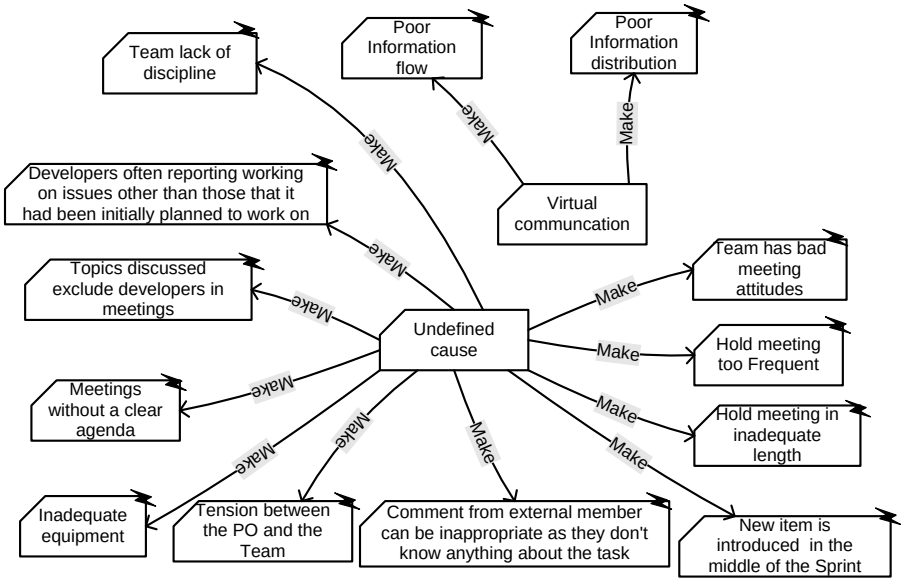


Fig. 8.19 Causes and Problems in *Daily meeting* visualized in iStar 2.0

8.4.4 Solving vulnerabilities

After identifying the vulnerabilities, we need to define necessary tasks that team members need to perform to avoid or solve them.

To avoid the vulnerabilities that may happen while performing *Daily Meeting*, the *Scrum Master* needs to help the *Development Team* understand its responsibilities and how to perform the tasks correctly. *Development Team* needs to take its responsibilities seriously and accomplish the assigned tasks.

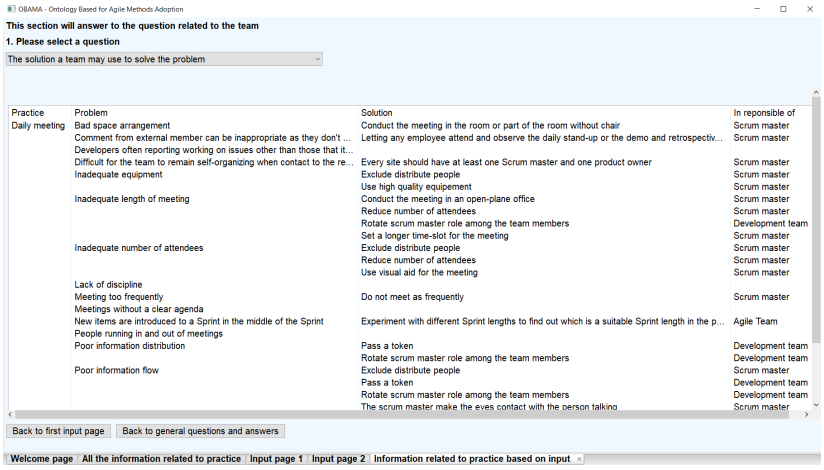


Fig. 8.20 Problems in *Daily meeting*, Solution and Role listed by OBAMA-Tool

To avoid the vulnerabilities found in the literature, practitioners can use the solutions provided by our tool, in the concern *the solution a team may use to solve the problem*. Figure 8.20 shows a list of the solutions to solve the problems. Based on this information, we add solutions to the previous diagram Figure 8.21 visualizes the causes, problems, and solutions. Some given solutions are however excluded as they are not applicable in the team's situation. For instance, the solution "Pass token" cannot be used to solve the "Poor information flow" problem in a team that communicates virtually. This diagram allows practitioners to brainstorm easily and identify what needs to be done to avoid or to solve the problems. For instance, we propose "Define clear rules for the meeting", where we represent in a gray hexagon, as a new solution to solve some problems as in Figure 8.19. Within the same concern, *the solution a team may use to solve the problem*, practitioners can also find the roles in charge for each proposed solution.

After identifying all the extra tasks to avoid the vulnerabilities, we can visualize the dependencies between roles as shown in Figure 8.22. As most of the problems are related to the organization of the meeting, the *Scrum Master* is responsible for most of the tasks while the *Development Team* is responsible for only a few of them. One task must be undertaken by both the *Scrum Master* and the *Development Team* together, also known as the *Scrum team*: find the suitable sprint length for the development. At the same time, the *Product Owner* must be responsible for handling conflicts and expectations to avoid tension for the team.

8.4.5 Implementing Agile Practice

After all the analysis done in the previous steps, practitioners can start implementing the practices into their development process. During the implementation, practitioners can use the diagrams created in the previous steps to keep track of things to be done for successful adoption. For instance, Figure 8.14

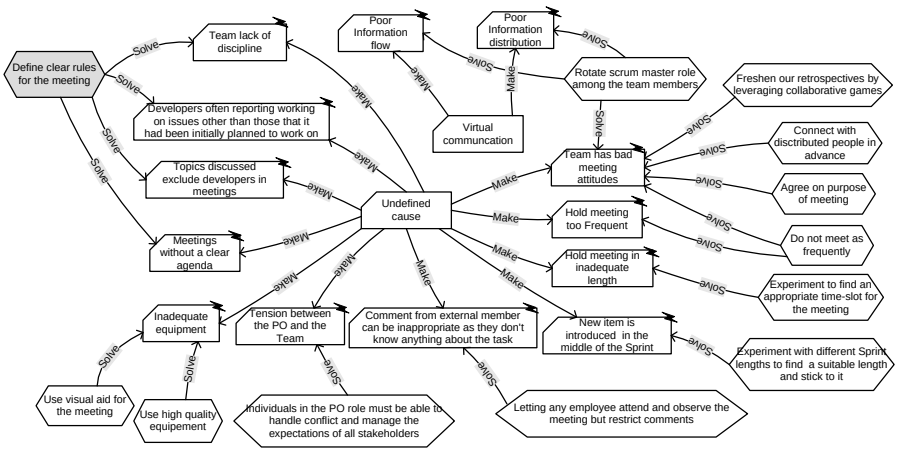


Fig. 8.21 Cause, Problems in *Daily meeting* and Solutions visualized in iStar 2.0

Fig. 8.22 Dependencies between roles to avoid vulnerabilities in *Daily meeting* visualized in iStar 2.0.

denotes what are the good situational factors to keep and what needs to be

improved. Each role in the team needs to accomplish the activities they are in charge of, as shown in Figure 8.17. The same diagram can also be used to trace and identify what activities the team may have failed to perform when confronted with problems and vice versa. Figure 8.21 is used to brainstorm for the possible problems and solutions. Finally, in addition to the activities which are parts of the practice, the team needs to perform some extra tasks to avoid the vulnerabilities as shown in Figure 8.22.

8.5 Conclusion

This chapter introduced a new framework for tailoring agile methods for adoption through the lens of socio-intentional dimensions. The first contribution in this chapter is the use of modeling techniques iStar 2.0 to ease the analysis process. Modeling allows visualizing relevant information altogether and allows team members to communicate, discuss and understand better how they should work together to successfully adopt agile practices. The second contribution is the methodology for tailoring agile methods that focuses on the why and who dimensions. With the help of OBAMA-Tool and iStar 2.0, our framework provides a methodology to analyze agile practices prior to the adoption in two levels: Tactical and Operational. By following our approach, practitioners can define the right strategies to achieve their goal, check their suitability for the adoption, define how to coordinate the activities of the various actors and how they depend on each other. In addition, each role can also understand the motivations and rationale behind the activities it has to perform. In the last contribution, to show how the framework works, we took an example of a real software Development Team. In step 1, we identified that "Daily Meeting", "Short Iteration" and "Sprint Planning" are the practices the team should adopt to achieve their goals. By choosing to focus on "Daily Meeting", in step 2, we emphasized that some situational factors have negative impacts on adoption. It is however still possible to adopt this practice since many other situational factors have positive impacts. In step 3, we identified how team members should perform their activities and depend on each other as they adopt a practice. At the same time, we identified vulnerabilities during the adoption caused by some roles and some possible problems found in the literature. In step 4, we pointed out the solutions to avoid the vulnerabilities. Finally, in step 5, we concluded what team members should do to successfully implement the practice in the development process.

The framework presents some limitations. First, the information provided by the tool is still limited. For instance, in step 3, as no information about the purpose of each activity is available, practitioners are required to understand it themselves to build the dependency diagram. In addition, some problems and solutions are not applicable to team's situations. Although our tool was built based on reliable evidence collected from empirical studies that had been reviewed and published, the underlying premise for this framework is however to focus on the methodology rather than the information provided by the tool. By this means, practitioners can customize the information or/and use other reliable and applicable sources of information accordingly. Second, this

framework is only dedicated to the planning part by focusing on how to prepare a team prior to agile practice adoption and a small part of the implementation. A complete framework should also include the evaluation process to further check the adoption success. Finally, it lacks an evaluation of how helpful this framework is in adopting the agile practice.

Part IV

Conclusion

Chapter 9

Conclusions

In this thesis, we studied the problem of Agile methods tailoring from a socio-intentional perspective. The purpose is to propose a framework that focuses on both social and intentional aspects in the agile methods tailoring process. We started by providing a general literature review of agile methodologies, and approaches of agile methods tailoring. Then, we reviewed some modeling frameworks related to the goal and social dimensions. This literature review provides us a general overview and a solid understanding of the subject. We then continued to find the concepts related to agile methods tailoring and understand their relationships based on the literature. In addition to a social-intentional conceptual model, we created an ontology and a supporting tool that make the information about agile methods adoption systematically reusable. Finally, we proposed a socio-intentional framework for agile methods tailoring which can guide practitioners to find the right practice to achieve their goals and to successfully integrate it into their development process.

In the intentional perspective, agile values and principles are known to be the core mindset or philosophy behind the creation of agile methodologies. These values and principles that were defined during the Agile Manifesto should be used as one of the main criteria for agile practices selection. However, we have observed that agile practitioners have been ignoring the Agile Manifesto and none of the agile values or principles is counted as selection criteria. To study and verify the relationships between the Agile Manifesto (value and principle) and agile practices selection, we conducted an SLR to extract problems, expectations, and the benefits behind agile methods or practice selection and compared them with agile values and principles. Our findings suggest that they are highly relevant. These correlations allow us to validate that the Agile Manifesto is highly relevant to agile method tailoring and it should be a criterion for agile practice selection.

Information and feedback on how agile practices have been adopted can vastly be found in academia and industrial knowledge bases. Such a collective experience allows the development of many approaches to simplify the adoption process and maximize the chances of success. With many approaches which have been proposed to guide and to help practitioners find the right practice that suits their team, none of them is systematic. Even though the knowledge is helpful for practitioners in tailoring agile practices fitting their team's situations,

it takes time and effort to find and locate relevant information as the data is unstructured and hardly exploitable. To make the knowledge of the previous empirical studies easily accessible, we conducted another SLR and exhaustively extracted 86 case studies on agile practice adoption. Using these case studies, we created an ontology to support knowledge representation about agile practice adoption. We then built a user-friendly prototype-tool, named OBAMA - an Ontology-Based tool for Agile Methods Adoption, that allows practitioners to efficiently retrieve the information.

In the social aspect, people, interactions and collaborations are known as one of the foundations of agile methods. In many case studies relating to agile practice adoption, they usually describe how to customize the agile practice in a given situation so that team members can work collaboratively. Even though many researches have shown that social is an important factor in the success of agile methods adoption, there are not many tailoring approaches that focus on this aspect. To address the lack of the social aspect in agile methods tailoring and to make the most of the existing knowledge, we proposed a socio-intentional framework. In our framework, we propose a methodology to analyze the suitability and vulnerability of agile practices and guide them on what they should do to successfully adopt the agile practice. Every step in the process of analyzing agile practice is done with the help of our prototype-tool and modeling techniques. Our tool is used to easily get useful information needed for analyzing agile practices based on real-life experiences while iStar 2.0 is used to visualize the information for a better understanding and to ease the analyzing process.

Hereafter, we summarize our contribution, limitations and some propositions for future works.

9.1 Summary of contributions

9.1.1 Validation of the Relationship between Agile Manifesto and Agile Practice Selection

The first contribution of this thesis is the validation of the relationship between the Agile Manifesto and agile practice selection. The result of this finding allows us to clarify another important factor that affects agile practice selection from a goal perspective. To this end, a systematic literature review (SLR) has been carried out to answer two research questions including: *how has the Agile Manifesto and its importance been discussed in tailored agile methods adoption?* and *is the Agile Manifesto related to agile practices selection?* In our SLR, we only consider formal data sources including IEEEXplorer, ScienceDirect, SpringerLink, and ACM Digital Library. The paper selection was done in 3 steps (early selection, abstract-based selection, and full-text screening selection) where each step has well-defined selection criteria. As result, 51 papers were selected for data extraction. Each paper was read carefully before we extracted relevant information.

Based on the results, we can conclude that *the Agile Manifesto has really lost attention from the development teams.* There are 51% of the selected

papers that talk about the Agile Manifesto while only 21% acknowledge its influence. On the contrary, by comparing the 4 values and 12 principles of the Agile Manifesto with the team's problems, expectations, and benefits extracted from the literature, at least 80% of them can be mapped to each other. This result allows us to conclude the second question that *the Agile Manifesto is highly relevant to the reasons behind the agile practice adoption.* As it still covers fundamental aspects of any agile method, it is important for the team to understand the Agile Manifesto before adopting any agile method and it should thus be a criterion for agile practice selection. For instance, starting from the team's problems, we can identify the relevant agile values/principles, and subsequently which agile practices to adopt. We hope our findings will encourage practitioners to give more attention to the Agile Manifesto before their adoption. For the research community, these results provide a clear-cut validation on the relation between the Agile Manifesto and agile practices, instead of being just an assumption or belief. This validation can be used as evidence to propose any approach or framework that can help improve agile practice adoption.

9.1.2 Ontology to Systematically Recycle Agile Practice Adoption Experiences

The second contribution of this thesis is an ontology that makes the existing knowledge about agile practices adoption reusable in a systematic manner. The procedure of ontology creation and validation requires a large data set to ensure that the ontology can represent the knowledge that will serve our future needs. We thus started by conducting another SLR to collect the knowledge and experience on agile practice adoption reported in the literature. We followed the same procedure we had used to conduct the previous SLR. As result, we exhaustively extracted 86 case studies on agile practice adoption experience from 79 individual selected papers. These case studies were then split into two sets: 10 studies were used as the training set while the other 76 studies were used as the validation set. Based on the knowledge from two main resources [84, 58] and data extracted from 10 case studies, we created an ontology to represent the knowledge about agile practice adoption. Next, we added seventeen inference rules to systematically discover more relationships among concepts in the ontology. Finally, we theoretically validated our ontology by following the corpus-based approach [21] and empirically validated it by using a survey with agile experts.

The result of the theoretical validation shows that our *ontology represents very well the information related to the agile practice adoption with minimum refinement for the unseen knowledge in the future.* The results from the survey with agile experts have shown that *our ontology can provide information efficiently and effectively.* In addition, *it helps a team decide if they should adopt a practice or not.* Based on these validation results, we can confidently say that we have built an ontology to recycle the useful knowledge about agile practice adoption. Our ontology model alone allows practitioners to quickly discover the useful concepts related to agile practices and their relationships. Practitioners can also use it to gain knowledge about agile practices in dept, to find the right

practice for their team, and to avoid the risk of failure in adoption. Researchers can use it to discover or/and understand different aspects of agile practice. This work has also shown how to use an ontology as an effective solution to recycle and to share knowledge among practitioners in a structured and exploitable way.

9.1.3 Evidence-based Tool

The third contribution of the thesis is a user-friendly tool that allows practitioners to retrieve the information from the ontology without any experience or knowledge required. Our prototype-tool, named *OBAMA - Ontology-Based tool for Agile Methods Adoption*, was written in Python programming language. Besides Python, there are three other main components needed for the development of our prototype-tool. That includes ontology file, owlready2, and wxPython. Our tool was created in a notebook-style where each page serves for a functionality. Using our tool, users can easily access all the information that we have inserted. By simply choosing one of the concerns from the combo-box list, the relevant information will be displayed underneath the list in a table format. Users can also filter for more specific information by providing their goals and team's situations in our input pages. To understand the advantages/disadvantages of our tool, we gathered expert's opinions through the same survey we used to validate the ontology.

The results show that *our tool can provide the information efficiently, effectively and it helps a team decide if they should adopt a practice*. Even though our tool is not fully satisfying for experts, they agree that it is good enough to serve our purpose. These results allow us to know that our tool is efficient for the users and we achieve our objective by using the tool to help practitioners understand agile practices.

9.1.4 Socio-intentional Framework for Agile Methods Tailoring

In the last contribution of this thesis, we proposed a socio-intentional framework that focuses on the intentional (why?) and social (who?) dimensions. In this framework, we propose a methodology to analyze agile practices and to define the strategies the team members should work together to successfully adopt a practice based on the team's goals, their situations, and dependencies between team members. In addition to the *intentional* and *social* dimensions, the proposed methodology also addresses most of the important concerns related to agile practice adoption found in the literature. In other words, in every step in the process of analyzing agile practice, practitioners can use our OBAMA-tool to easily get the needed information. As it is hard to analyze the information listed by the tool in textual format, we proposed to represent the information in graphical models before the analysis. To do that, we identified suitable modeling language and notions by mapping the concepts and relationships in our ontology model with the notions of different modeling frameworks that can represent either goal or social dimension. As result, iStar 2.0 was chosen to use in our framework as its notions match best with our agile elements. For the agile concepts that cannot be mapped, we propose extended notions for the

representation. Finally, we demonstrated how to use our framework by applying our framework to a case of a real software development project.

The result of our framework allows practitioners to customize agile methods to fit their goals and situations. It provides a methodology to analyze agile practitioners in two levels. *Tactical level* allows practitioners to (1) identify the right practices to adopt based on their goals and (2) check the suitability of a team based on their situations. *Operational level* allows the team to identify (1) the vulnerabilities during the practice adoption caused by team members, (2) possible problems based on the experiences, and (3) solutions to avoid the vulnerabilities and solve the problems. In the whole analyzing process, every step is done more efficiently with the help of our tool and modeling techniques. Using the tool can help practitioners get the needed information easily and also make the most out of the existing experiences. The analysis by means of graphical models helps practitioners to communicate, discuss and understand better how they should work together to successfully adopt agile practices.

9.2 Limitations and Future Works

9.2.1 Ontology Model and Knowledge

Our ontology model and knowledge always need to be expanded using data from both the literature and real-life case studies. The currently available knowledge in our ontology is only related to the five most commonly-used practices collected from formal resources through an SLR approach. This amount is still small compared to the number of existing agile practices. There are four main approaches we can do to expand its knowledge.

- We continue our SLR to collect the knowledge related to other commonly-used practices such as release planning, planning poker, kanban, pair programming, etc;
- We collect the knowledge from informal sources such as websites, blogs, forums. The reason is the fact that it is more practical to share knowledge on such platforms. Many famous authors including the Agile Manifesto authors have been sharing their knowledge through informal platforms. To do that, we need to have well-defined criteria for selection that can ensure that the extracted knowledge is reliable, e.g., the credibility of the platform, author, article, etc;
- Apart from the literature, we need to extract data through empirical research such as observation or in-depth interviews with the members of real software development teams. We believe that there is valuable knowledge that cannot be textually described but perceived by observation;
- We put our ontology as open-source so that it can be reused, improved, and expanded by the agile community.

9.2.2 Evidence-based Tool

Our tool is at the prototype stage and it only means to simplify the information retrieval process. There are parts of it that need to be improved and developed in both usability and functionality.

- In the current version of our tool, the only way practitioners can get the information is by selecting the concerns from a combo-box list. A more sophisticated tool would allow users to access more easily from one concept or concern to another. For instance, when the users right-click on a *problem*, they should be able to choose what other related concern they want to access next, such as *causes* or *solutions*;
- The information is listed in a table format with a fixed row side. To display long information, the users need to manually expand the column size. It would be more convenient for users to read if the information is wrapped into multiple lines to fit the column size;
- The information listed by the tool was cut short and straight to the point while the users sometimes need context to fully understand it. The tool thus needs to have a functionality that allows users to find further information or the original sources;
- In addition to retrieving and displaying the information, the tool also needs to have a functionality that allows the users to encode new knowledge easily. To facilitate the encoding process, our tool should suggest step by step the classes of the instance that users need to add and then how to build the relationship between instances;
- The tool should also include the functionality that allows users to evaluate the credibility of the existing knowledge for instance by rating or commenting.

9.2.3 Socio-intentional Framework for Agile Methods Tailoring

Our framework presents some limitations related to the methodology for tailoring, the usage of the tool, and the modeling techniques.

- The knowledge provided by the tool is still limited. Practitioners cannot always count on our tool to get all the information needed for the analysis. At some steps, practitioners are required to analyze the practice based on their understanding;
- As there is no information that can be one-size-fits-all, there can be some conflicts between the information provided in the tool. In such case, practitioners need to decide which piece of information suits best to their needs and situations;
- The explanation on how to model the information provided by the tool in iStar 2.0 can help practitioners to build a model even without any knowledge of this modeling language. It however still takes time and

effort to do it. This process could be simpler if the tool can automatically generate the selected information in the tool into iStar 2.0 elements;

- We have applied a case of a real software development team but there is no evaluation on this framework. To fully understand the usefulness of the framework, we need to conduct more sophisticated empirical research with a real software development team. The research method is observation and it aims at evaluating two points: (1) the behavior of the team members during agile practice adoption while following our framework and (2) the effectiveness of the framework. The former aims at evaluating if team members improve the understanding of the agile practice and their roles during the agile practice implementation. The latter aims at evaluating if the suggestions of our framework are correct and if it can help the practitioners to successfully adopt the agile practice and eventually achieve their goals;

- Our framework has not been designed for practice adoption in the context of scaled agile methods adoption (like the Scaled Agile Framework (SAFe) [95] for example). Even though our tool contains information about how to adopt agile practices within the context of large and distributed teams, scaled agile methods do use a completely different approach including a heavy and complex hierarchy. We could however extend our framework to deal with such methods. To achieve that, we first need to extract the important concepts and relationships architecting scaled agile frameworks. Based on these, we need to identify the necessary extra steps for our framework to fit with scalable agility. For instance, at the tactical level, we need to propose how to build a strategic plan that covers the goals in different layers, how to align these goals, and what are the right practices to achieve them. At the operational level, we need to address how to manage multiple teams to work in parallel on the parts of the same project, how to align schedules and avoid conflicts, how to integrate the results, etc;

- This framework is dedicated mostly to the pre-adoption stage of agile methods adoption. 4 among 5 steps of our methodology focus on how to prepare a team prior to agile practice adoption where one last part focuses on the implementation. To make the framework complete, we need to also include the post-adoption and evaluation process to further check the adoption success and measure the agile maturity level of the team.

References

[1] (2004). Daml ontology library. http://www.daml.org/ontologies.

[2] (2018). Protege ontology library. https://protegewiki.stanford.edu/wiki/Protege_Ontology_Library.

[3] (2019). Inference. https://www.w3.org/standards/semanticweb/inference.

[4] Abbas, N., Gravell, A. M., and Wills, G. B. (2008). Historical roots of agile methods: Where did "agile thinking" come from? In *International conference on agile processes and extreme programming in software engineering*, pages 94–103. Springer.

[5] Abbas, N., Gravell, A. M., and Wills, G. B. (2010). Using factor analysis to generate clusters of agile practices (a guide for agile process improvement). In *AGILE Conference, 2010*, pages 11–20. IEEE.

[6] Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J. (2017). Agile software development methods: Review and analysis. *arXiv preprint arXiv:1709.08439*.

[7] AgileAlliance (2005a). Kanban. https://www.agilealliance.org/glossary/kanban.

[8] AgileAlliance (2005b). Scrum. https://www.agilealliance.org/glossary/scrum.

[9] Ahmed, E.-M. and Sidky, A. (2009). 25 percent ahead of schedule and just at "step 2" of the sami. In *Agile Conference, 2009. AGILE'09.*, pages 162–169. IEEE.

[10] Ambler, S. (2005). The Agile Unified Process (AUP). *Ambysoft, http://www.agilealliance. hu/materials/books/SWA-AUP. pdf*.

[11] Ambler, S. W. and Lines, M. (2016). The disciplined agile process decision framework. In *International Conference on Software Quality*, pages 3–14. Springer.

[12] Auvinen, J., Back, R., Heidenberg, J., Hirkman, P., and Milovanov, L. (2006). Software process improvement with agile practices in a large telecom company. In *International Conference on Product Focused Software Process Improvement*, pages 79–93. Springer.

[13] Avison, D. E. and Wood-Harper, A. T. (1991). Information systems development research: an exploration of ideas in practice. *The Computer Journal*, 34(2):98–112.

[14] Ayed, H., Vanderose, B., and Habra, N. (2012). A metamodel-based approach for customizing and assessing agile methods. In *Quality of Information and Communications Technology (QUATIC)*, pages 66–74. IEEE.

[15] Baleviciute, G. (2014). Whitepaper–scrum vs kanban vs. scrumban. *Retrieved September*, 10:2016.

[16] Bass, J. M. (2014). Scrum master activities: process tailoring in large enterprise projects. In *Global Software Engineering (ICGSE), 2014 IEEE 9th International Conference on*, pages 6–15. IEEE.

[17] Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[18] Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). Manifesto for agile software development.

[19] Benyon, D. and Skidmore, S. (1987). Towards a tool kit for the systems analyst. *The computer journal*, 30(1):2–7.

[20] Berteig, M. (2008). Experience report: Extremely short iterations as a catalyst for effective prioritization of work. In *Agile, 2008. AGILE'08. Conference*, pages 265–268. IEEE.

[21] Bishop, C. M. (2007). *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer.

[22] Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5):61–72.

[23] Bogojević, P. (2017). Comparative analysis of agile methods for managing software projects. *European Project Management Journal*, 7(1):58–74.

[24] Booch, G., Rumbaugh, J., and Jacobson, I. (2005). Unified modeling language user guide,(the 2nd edition).

[25] Boone, H. N. and Boone, D. A. (2012). Analyzing likert data. *Journal of extension*, 50(2):1–5. https://www.joe.org/joe/2012april/tt2.php.

[26] Bowers, J., May, J., Melander, E., Baarman, M., and Ayoob, A. (2002). Tailoring xp for large system mission critical software development. In *Conference on Extreme Programming and Agile Methods*, pages 100–111. Springer.

[27] Brank, J., Grobelnik, M., and Mladenic, D. (2005). A survey of ontology evaluation techniques. In *Proceedings of the conference on data mining and data warehouses (SiKDD 2005)*, pages 166–170. Citeseer Ljubljana, Slovenia.

[28] Brewster, C., Alani, H., Dasmahapatra, S., and Wilks, Y. (2004). Data driven ontology evaluation.

[29] Burnham, K. P. and Anderson, D. R. (2004). Multimodel inference: understanding aic and bic in model selection. *Sociological methods & research*, 33(2):261–304.

[30] Burrows, M. (2014). *Kanban from the Inside*. Sequim, WA: Blue Hole Press.

[31] Campanelli, A. S. (2014). A model for agile method tailoring. *Projetos e Dissertações em Sistemas de Informação e Gestão do Conhecimento*, 3(2).

[32] Campanelli, A. S. and Parreiras, F. S. (2015). Agile methods tailoring–a systematic literature review. *Journal of Systems and Software*, 110:85–100.

[33] Castro, J., Kolp, M., and Mylopoulos, J. (2001). A requirements-driven development methodology. In *International Conference on Advanced Information Systems Engineering*, pages 108–123. Springer.

[34] Chandra, C. and Tumanyan, A. (2007). Organization and problem ontology for supply chain information support system. *Data & Knowledge Engineering*, 61(2):263–280.

[35] Chandrasekaran, B., Josephson, J. R., and Benjamins, V. R. (1999). What are ontologies, and why do we need them? *IEEE Intelligent Systems and their applications*, 14(1):20–26.

[36] Chen, Y.-J. (2010). Development of a method for ontology-based empirical knowledge representation and reasoning. *Decision Support Systems*, 50(1):1–20.

[37] Chung, L., Nixon, B. A., Yu, E., and Mylopoulos, J. (2000). The nfr framework in action. In *Non-Functional Requirements in software engineering*, pages 15–45. Springer.

[38] Cockburn, A. (1998). *Surviving Object-oriented Projects: A Manager's Guide*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[39] Cockburn, A. (2004). *Crystal clear: a human-powered methodology for small teams*. Pearson Education.

[40] Cohen, D., Lindvall, M., and Costa, P. (2004). An introduction to agile methods. *Advances in computers*, 62(03):1–66.

[41] Conboy, K. and Fitzgerald, B. (2010). Method and developer characteristics for effective agile method tailoring: A study of xp expert opinion. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 20(1):1–30.

[42] Dalpiaz, F., Franch, X., and Horkoff, J. (2016). istar 2.0 language guide. *arXiv preprint arXiv:1605.07767*.

[43] Damiani, E., Colombo, A., Frati, F., and Bellettini, C. (2007). A metamodel for modeling and measuring scrum development process. In *Agile Processes in Software Engineering and Extreme Programming, 8th International Conference, XP 2007, Proceedings*, pages 74–83.

[44] Dardenne, A., Van Lamsweerde, A., and Fickas, S. (1993). Goal-directed requirements acquisition. *Science of computer programming*, 20(1-2):3–50.

[45] Davis, G. B. (1982). Strategies for information requirements determination. *IBM systems journal*, 21(1):4–30.

[46] de Azevedo Santos, M., de Souza Bermejo, P. H., de Oliveira, M. S., Tonelli, A. O., et al. (2011). Agile practices: An assessment of perception of value of professionals on the quality criteria in performance of projects. *Journal of Software Engineering and Applications*, 4(12):700.

[47] Denning, P. J. (1997). A new social contract for research. *Communications of the ACM*, 40(2):132–134.

[48] Derbier, G. (2003). Agile development in the old economy. In *Agile Development Conference, 2003. ADC 2003. Proceedings of the*, pages 125–131. IEEE.

[49] Diebold, P. and Zehler, T. (2016). The right degree of agility in rich processes. In *Managing Software Process Evolution*, pages 15–37. Springer.

[50] Eckstein, J. (2013). *Agile software development in the large: Diving into the deep*. Addison-Wesley.

[51] Eilers, K., Simmert, B., and Peters, C. (2020). Doing agile vs. being agile-understanding their effects to improve agile work.

[52] Ejigu, D., Scuturici, M., and Brunie, L. (2007). An ontology-based approach to context modeling and reasoning in pervasive computing. In *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07)*, pages 14–19. IEEE.

[53] Eloranta, V.-P., Koskimies, K., and Mikkonen, T. (2016). Exploring scrumbut—an empirical study of scrum anti-patterns. *Information and Software Technology*, 74:194–203.

[54] Eric, S. Y. (2009). Social modeling and i. In *Conceptual Modeling: Foundations and Applications*, pages 99–121. Springer.

[55] Erickson, J., Lyytinen, K., and Siau, K. (2005). Agile modeling, agile software development, and extreme programming: the state of research. *Journal of database Management*, 16(4):88.

[56] Esfahani, H. C., Cabot, J., and Yu, E. (2010a). Adopting agile methods: Can goal-oriented social modeling help? In *Research Challenges in Information Science (RCIS), 2010 Fourth International Conference on*, pages 223–234. IEEE.

[57] Esfahani, H. C., Eric, S., and Annosi, M. C. (2011). Towards the strategic analysis of agile practices. In *CAiSE Forum*, pages 155–162.

[58] Esfahani, H. C. and Yu, E. (2010). A repository of agile method fragments. In *International Conference on Software Process*, pages 163–174. Springer.

[59] Esfahani, H. C., Yu, E., and Cabot, J. (2010b). Situational evaluation of method fragments: An evidence-based goal-oriented approach. In *International Conference on Advanced Information Systems Engineering*, pages 424–438. Springer.

[60] Fitzgerald, B., Hartnett, G., and Conboy, K. (2006). Customising agile methods to software practices at intel shannon. *European Journal of Information Systems*, 15(2):200–213.

[61] Fitzgerald, B., Russo, N., and O'Kane, T. (2000). An empirical study of system development method tailoring in practice. *ECIS 2000 Proceedings*, page 4.

[62] Flora, H. K. and Chande, S. V. (2014). A systematic study on agile software development methodologies and practices. *International Journal of Computer Science and Information Technologies*, 5(3):3626–3637.

[63] Florac, W. A. and Carleton, A. D. (1999). *Measuring the software process: statistical process control for software process improvement*. Addison-Wesley Professional.

[64] Forsberg, K. and Mooz, H. (1991). The relationship of system engineering to the project cycle. In *INCOSE International Symposium*, volume 1, pages 57–65. Wiley Online Library.

[65] Forte, F. and Kloppenborg, T. (2018). The agile mindset for project management. In *International Research Network on Organizing by Projects (IRNOP) 2017*, pages 1–15. UTS ePRESS, Sydney.

[66] Fowler, M., Highsmith, J., et al. (2001). The agile manifesto. *Software Development*, 9(8):28–35.

[67] Franch, X., López, L., Cares, C., and Colomer, D. (2016). The i* framework for goal-oriented modeling. In *Domain-specific conceptual modeling*, pages 485–506. Springer.

[68] Gregorio, D. D. (2012). How the business analyst supports and encourages collaboration on agile projects. In *Systems Conference (SysCon), 2012 IEEE International*, pages 1–4. IEEE.

[69] Gremillion, L. and Pyburn, P. (1986). Breaking the systems development bottleneck. *Harvard Business Review (March/April 1983)*.

[70] Harmsen, A. F., Brinkkemper, J. N., and Oei, J. H. (1994). *Situational method engineering for information system project approaches*. Citeseer.

[71] Henderson-Sellers, B. and Gonzalez-Perez, C. (2005). A comparison of four process metamodels and the creation of a new generic standard. *Information and software technology*, 47(1):49–65.

[72] Henderson-Sellers, B. and Ralyté, J. (2010). Situational method engineering: state-of-the-art review. *Journal of Universal Computer Science*.

[73] Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, pages 75–105.

[74] Highsmith, J. (2013). *Adaptive software development: a collaborative approach to managing complex systems*. Addison-Wesley.

[75] Hummel, M. (2014). State-of-the-art: A systematic literature review on agile information systems development. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 4712–4721. IEEE.

[76] Jalali, S. and Wohlin, C. (2010). Agile practices in global software engineering-a systematic map. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*, pages 45–54. IEEE.

[77] Jalali, S. and Wohlin, C. (2012). Global software engineering and agile practices: a systematic review. *Journal of software: Evolution and Process*, 24(6):643–659.

[78] Kalus, G. and Kuhrmann, M. (2013). Criteria for software process tailoring: a systematic review. In *Proceedings of the 2013 International Conference on Software and System Process*, pages 171–180.

[79] Karlström, D. and Runeson, P. (2006). Integrating agile software development into stage-gate managed product development. *Empirical Software Engineering*, 11(2):203–225.

[80] Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26.

[81] Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., and Linkman, S. (2009). Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, 51(1):7–15.

[82] Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

[83] Kiv, S., Heng, S., Kolp, M., and Wautelet, Y. (2017a). An intentional perspective on partial agile adoption. In *Proceedings of the 12th International Conference on Software Technologies - Volume 1: ICSOFT,*, pages 116–127. INSTICC, SciTePress.

[84] Kiv, S., Heng, S., Kolp, M., and Wautelet, Y. (2018). Agile manifesto and practices selection for tailoring software development: A systematic literature review. In *International Conference on Product-Focused Software Process Improvement*, pages 12–30. Springer.

[85] Kiv, S., Heng, S., Kolp, M., and Wautelet, Y. (2019). Agile methods knowledge representation for systematic practices adoption. In *International Conference on Agile Software Development*, pages 19–34. Springer.

[86] Kiv, S., Heng, S., Wautelet, Y., and Kolp, M. (2017b). Towards a goal-oriented framework for partial agile adoption. In *Software Technologies - 12th International Joint Conference, ICSOFT 2017, Revised Selected Papers*, pages 69–90.

[87] Kniberg, H. and Skarin, M. (2010). *Kanban and Scrum-making the most of both*. Lulu. com.

[88] Kruchten, P. (2004). *The rational unified process: an introduction*. Addison-Wesley Professional.

[89] Kumar, K. and Welke, R. J. (1992). Methodology engineeringr: a proposal for situation-specific methodology construction. In *Challenges and strategies for research in systems development*, pages 257–269.

[90] Kurapati, N., Manyam, V. S. C., and Petersen, K. (2012). Agile software development practice adoption survey. In *International Conference on Agile Software Development*, pages 16–30. Springer.

[91] Lamy, J.-B. (2017). Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine*, 80:11–28.

[92] Larman, C. and Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer*, 36(6):47–56.

[93] Lawshe, C. H. (1975). A quantitative approach to content validity 1. *Personnel psychology*, 28(4):563–575.

[94] Lee, S. and Yong, H.-S. (2013). Agile software development framework in a small project environment. *Journal of Information Processing Systems*, 9(1):69–88.

[95] Leffingwell, D., Jemilo, D., Zamora, M., ONeill, C., and Yakuma, A. (2014). Scaled agile framework (safe). *Haettu*, 27:2014. https://www.scaledagileframework.com/.

[96] Lin, J., Miao, C., Shen, Z., and Sun, W. (2013). Goal oriented agile unified process (goaup): An educational case study. In *2013 International Conference on Software Engineering and Computer Science*, pages 36–44. Atlantis Press.

[97] Lin, J., Yu, H., Shen, Z., and Miao, C. (2014). Using goal net to model user stories in agile software development. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on*, pages 1–6. IEEE.

[98] Lycett, M., Patel, C., Merico, A., Iacovelli, N., and de Cesare, S. (2008). Tailoring software development methodologies in practice: A case study. *Journal of computing and information technology*, 16(3):157–168.

[99] Madeyski, L. (2010). *Test-Driven Development: An Empirical Evaluation of Agile Practice*. Springer Publishing Company, Incorporated, 1st edition.

[100] Madi, T., Dahalin, Z., and Baharom, F. (2011). Content analysis on agile values: A perception from software practitioners. In *Software Engineering (MySEC), 2011 5th Malaysian Conference in*, pages 423–428. IEEE.

[101] Maham, M. (2008). Planning and facilitating release retrospectives. In *Agile 2008 Conference*, pages 176–180. IEEE.

[102] March, S. T. and Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems*, 15(4):251–266.

[103] Martin, J. (1991). *Rapid application development*. Macmillan Publishing Co., Inc.

[104] McGuinness, D. L., Van Harmelen, F., et al. (2004). Owl web ontology language overview. *W3C recommendation*, 10(10):2004.

[105] Mikulėnas, G., Butleris, R., and Nemuraitė, L. (2011). An approach for the metamodel of the framework for a partial agile method adaptation. *Information Technology And Control*, 40(1):71–82.

[106] Moe, N. B. and Aurum, A. (2008). Understanding decision-making in agile software development: a case-study. In *Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference*, pages 216–223. IEEE.

[107] Moe, N. B., Aurum, A., and Dybå, T. (2012). Challenges of shared decision-making: A multiple case study of agile software development. *Information and Software Technology*, 54(8):853–865.

[108] Noy, N. F. and Hafner, C. D. (1997). The state of the art in ontology design: A survey and comparative review. *AI magazine*, 18(3):53–53.

[109] Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. https://protege.stanford.edu/publications/ontology_development/ontology101.pdf.

[110] Object Managment Group (2008). Software systems process engineering metamodel. https://www.omg.org/spec/SPEM.

[111] Ochodek, M. and Kopczyńska, S. (2018). Perceived importance of agile requirements engineering practices–a survey. *Journal of Systems and Software*, 143:29–43.

[112] Paasivaara, M. and Lassenius, C. (2016). Scaling scrum in a large globally distributed organization: a case study. In *Global Software Engineering (ICGSE), 2016 IEEE 11th International Conference on*, pages 74–83. IEEE.

[113] Palmer, S. R. and Felsing, M. (2001a). *A Practical Guide to Feature-Driven Development*. Pearson Education, 1st edition.

[114] Palmer, S. R. and Felsing, M. (2001b). *A practical guide to feature-driven development*. Pearson Education.

[115] Pereira, T., Alencar, F. M., and Castro, J. (2016). Bvccon-tool: A modeling tool to support dynamic business process configuration approach. In *CIbSE*, pages 39–52.

[116] Poppendieck, M. and Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit: An Agile Toolkit*. Addison-Wesley.

[117] Potoniec, J., Wiśniewski, D., Ławrynowicz, A., and Keet, C. M. (2020). Dataset of ontology competency questions to sparql-owl queries translations. *Data in Brief*, page 105098.

[118] Precord, C. (2015). *WxPython Application Development Cookbook*. Packt Publishing Ltd.

[119] Pressman, R. S. (2005). *Software engineering: a practitioner's approach*. Palgrave Macmillan.

[120] Prud'hommeaux, E. and Seaborne, A. (2005). Sparql query language for rdf http://www.w3.org. Technical report, TR/rdf-sparql-query. https://www.w3.org/TR/rdf-sparql-query/.

[121] Qumer, A. and Henderson-Sellers, B. (2008). A framework to support the evaluation, adoption and improvement of agile methods in practice. *Journal of Systems and Software*, 81(11):1899–1919.

[122] Raad, J. and Cruz, C. (2015). A survey on ontology evaluation methods.

[123] Rahman, A., Agrawal, A., Krishna, R., Sobran, A., and Menzies, T. (2018). "doing" agile versus "being" agile.

[124] Rao, L., Mansingh, G., and Osei-Bryson, K.-M. (2012). Building ontology based knowledge maps to assist business process re-engineering. *Decision Support Systems*, 52(3):577–589.

[125] Rao, L., Reichgelt, H., and Osei-Bryson, K.-M. (2009). An approach for ontology development and assessment using a quality framework. *Knowledge Management Research & Practice*, 7(3):260–276.

[126] Reddy, A. (2015). *The Scrumban [r] evolution: getting the most out of Agile, Scrum, and lean Kanban*. Addison-Wesley Professional.

[127] Respect, I. (2007). A kaos tutorial.

[128] Royce, W. (1970). The software lifecycle model (waterfall model). In *Proc. Westcon*, volume 314.

[129] Saleh, M. H. (2013). *Methodology for selection of agile practices*. PhD thesis.

[130] Santos, R., Flentge, F., Begin, M.-E., and Navarro, V. (2011). Agile technical management of industrial contracts: Scrum development of ground segment software at the european space agency. In *International Conference on Agile Software Development*, pages 290–305. Springer.

[131] Schmidt, M., Meier, M., and Lausen, G. (2010). Foundations of sparql query optimization. In *Proceedings of the 13th International Conference on Database Theory*, pages 4–33.

[132] Schön, E.-M., Thomaschewski, J., and Escalona, M. J. (2017). Agile requirements engineering: A systematic literature review. *Computer Standards & Interfaces*, 49:79–91.

[133] Schuppenies, R. and Steinhauer, S. (2002). Software process engineering metamodel. *OMG group, November*.

[134] Schwaber, K. (1996). Controlled chaos: Living on the edge. *American Programmer*, 9:10–16.

[135] Schwaber, K. (2004). *Agile project management with Scrum*. Microsoft press.

[136] Schwaber, K. and Beedle, M. (2002). *Agile software development with Scrum*, volume 1. Prentice Hall.

[137] Scrum study (2017). Why scrum - scrum principles. https://www.scrumstudy.com/whyscrum/scrum-principles.

[138] Séguin, N., Tremblay, G., and Bagane, H. (2012). Agile principles as software engineering principles: An analysis. In *International Conference on Agile Software Development*, pages 1–15. Springer.

[139] Shen, Z., Miao, C., Tao, X., and Gay, R. (2004). Goal oriented modeling for intelligent software agents. In *Intelligent Agent Technology, 2004.(IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, pages 540–543. IEEE.

[140] Shu, X., Turinsky, A., Sensen, C., and Maurer, F. (2007). A case study of the implementation of agile methods in a bioinformatics project. In *Proceedings of the 8th international conference on Agile processes in software engineering and extreme programming*, pages 169–170. Springer-Verlag.

[141] Sidky, A. S., Arthur, J. D., and Bohner, S. A. (2007). A disciplined approach to adopting agile practices: the agile adoption framework. *ISSE*, 3(3):203–216.

[142] Söderström, E., Andersson, B., Johannesson, P., Perjons, E., and Wangler, B. (2002). Towards a framework for comparing process modelling languages. In *International Conference on Advanced Information Systems Engineering*, pages 600–611. Springer.

[143] Sommerville, I. (2011). Software engineering 9th edition. *ISBN-10*, 137035152:18.

[144] Stapleton, J. (1997a). *DSDM, dynamic systems development method: the method in practice*. Cambridge University Press.

[145] Stapleton, J. (1997b). *Dsdm: The Method in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[146] StateOfAgile (2019). The 13th Annual State of Agile Report. https://www.stateofagile.com.

[147] Straub, D., Boudreau, M.-C., and Gefen, D. (2004). Validation guidelines for is positivist research. *Communications of the Association for Information systems*, 13(1):24.

[148] Stray, V. G., Lindsjorn, Y., and Sjoberg, D. I. (2013). Obstacles to efficient daily meetings in agile development projects: A case study. In *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*, pages 95–102. IEEE.

[149] Susi, A., Perini, A., Mylopoulos, J., and Gi, P. (2005). The tropos metamodel and its use. *Informatica*, 29(4).

[150] Sutherland, J. (2010). Agile principles and values. *Recuperado de: http://msdn. microsoft. com/en-us/library/dd997578. aspx*.

[151] Sutherland, J. and Schwaber, K. (2013). The scrum guide. *The definitive guide to scrum: The rules of the game. Scrum. org*, 268.

[152] Taherdoost, H. (2016). Validity and reliability of the research instrument; how to test the validation of a questionnaire/survey in a research.

[153] Thomas, D. (2005). *Programming Ruby The Pragmatic Programmers' Guide*. he Pragmatic Bookshelf.

[154] Thomas, D. (2014). Agile is dead, https://pragdave.me/blog/2014/03/04/time-to-kill-agile.html.

[155] Tolvanen, J.-P. (1998). Incremental method engineering with modeling tools. *Jyväskylä Studies in Computer Science, Economics and Statistics*, 47.

[156] Tripp, J. F. and Armstrong, D. J. (2014). Exploring the relationship between organizational adoption motives and the tailoring of agile methods. In *47th Hawaii International Conference on System Sciences (HICSS)*, pages 4799–4806. IEEE.

[157] Tsichritzis, D. (1998). The dynamics of innovation in beyond calculation: The next fifty years of computing, pj denning and rm metcalfe. pages 259–265.

[158] Uschold, M. and Gruninger, M. (1996). Ontologies: Principles, methods and applications. *The knowledge engineering review*, 11(2):93–136.

[159] Van Kelle, E., Visser, J., Plaat, A., and van der Wijst, P. (2015). An empirical study into social success factors for agile software development. In *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 77–80. IEEE.

[160] VersionOne (2017). 11th annual state of agile development survey. https://stateofagile.com/#ufh-i-613554036-11th-annual-state-of-agile-report/7027494.

[161] VersionOne (2019). 13th annual state of agile survey. https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508.

[162] Wautelet, Y., Heng, S., Kiv, S., and Kolp, M. (2017). User-story driven development of multi-agent systems: A process fragment for agile methods. *Computer Languages, Systems & Structures*, 50:159–176.

[163] Wautelet, Y., Heng, S., Kolp, M., and Mirbel, I. (2014). Unifying and extending user story models. In *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014. Proceedings*, pages 211–225.

[164] Yu, E. (2011). Modeling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering*, 11(2011):66–87.

[165] Yu, E. and Mylopoulos, J. (1994). Understanding "why" in software process modelling, analysis, and design. In *Proc. of the 16th Int. Conf. on Software Engineering*, pages 159–168. IEEE Computer Society Press.

Appendix A

Ontology Model Validation: Survey Questions

Ontology-Based for Agile Methods Adoption (OBAMA) Tool

Welcome to the survey for an evaluation of a Ontology-Based for Agile Methods Adoption (OBAMA).

The validation will take place in two rounds (or three if necessary). In each round, you will receive a questionnaire. It will take approximately 30 minutes to complete each round.
- In the first round, we will ask you to indicate to what extent you agree with the statements related to the quality of our tool.
- The second round is to find a consensus amongst the expert about the changes in the tool. This time we will give feedback on the results of the first round to improve our tool.

*If we could not find the consensus after the second round, there will be a third round of the survey which will be similar to the second round.

During the survey, participant will be guided on how to use the tool in order to answer the questions.

Note: Your participation in this research is voluntary. You have the right to withdraw at any point during the study, for any reason, and without any prejudice.
The information provided will be used for the research purpose only. None of the information will be misused and all information will be kept confidential.

We will provide you with the final result of this validation exercise.

By clicking the button below, you acknowledge that your participation in the study in voluntary, you are 18 years of age and that you are aware that you may choose to terminate your participation in the study at any time and for any reason.

- ○ I consent, begin the study
- ○ I do not consent, I do not wish to participate

Name of institution

_____

What is your role(s) in your organization?

- ○ Agile coach
- ○ Project manager
- ○ Team leader
- ○ Scrum master
- ○ Software developer
- ○ Product owner
- ○ Software architecture
- ○ Administrator
- ○ Other _____

Please state your highest level of education, e.g. Master in Advance Software Engineering

_____

How long you have been working with agile methods?

- ○ < 1 year
- ○ 1 - 3 years
- ○ 3 - 5 years
- ○ 5 - 10 years
- ○ > 10 years

How long have your team been using agile methods?

- ○ < 1 year
- ○ 1 - 3 years
- ○ 3 - 5 years
- ○ 5 - 10 years
- ○ > 10 years

Do you own any agile certification?

- ○ Yes, I do
- ○ No, I do not

If you own any agile certification, please state below:

_____

_____

_____

QI.1 Consider your own experience with agile, how often do you ***need*** the information related to each concern below before you start adopting each agile practice?

	Never	Rare	Sometimes	Often	Always
C1. The objective/goal a team can achieve by adopting an agile practice	○	○	○	○	○
C2. The agile value a team can achieve by adopting a practice	○	○	○	○	○
C3. The agile principle a team can achieve by adopting a practice	○	○	○	○	○
C4. The activity a team should perform as part of a practice	○	○	○	○	○
C5. The problem a team may encounter while adopting a practice	○	○	○	○	○
C6. The situation of the team which is bad for adopting a practice	○	○	○	○	○
C7. The situation of the team which is good for adopting a practice	○	○	○	○	○
C8. The activity that they perform which is bad for adopting a practice	○	○	○	○	○
C9. The activity that they perform which is good for adopting a practice	○	○	○	○	○
C10. The artifact required for adopting a practice	○	○	○	○	○
C11. The role required for adopting a practice	○	○	○	○	○
C12. The requisites a team should prepare in order to successfully adopt a practice	○	○	○	○	○
C13. The cause of the problem team may encounter	○	○	○	○	○

C14. The solution a team may use to solve the problem	○	○	○	○	○
C15. The general knowledge based on experiences related to agile practice a team should learn	○	○	○	○	○

Please explain your answers to the above, concern by concern, if your answer is Never or Rare

_____

_____

_____

_____

_____

QI.2 How would you rate the ***relevancy level*** of each concern below to the agile practice adoption?

	Irrelevant	Useful to learn	Important to learn	Necessary to learn
C1. The objective/goal a team can achieve by adopting an agile practice	○	○	○	○
C2. The agile value a team can achieve by adopting a practice	○	○	○	○
C3. The agile principle a team can achieve by adopting a practice	○	○	○	○
C4. The activity a team should perform as part of a practice	○	○	○	○
C5. The problem a team may encounter while adopting a practice	○	○	○	○
C6. The situation of the team which is bad for adopting a practice	○	○	○	○
C7. The situation of the team which is good for adopting a practice	○	○	○	○
C8. The activity that they perform which is bad for adopting a practice	○	○	○	○
C9. The activity that they perform which is good for adopting a practice	○	○	○	○
C10. The artifact required for adopting a practice	○	○	○	○
C11. The role required for adopting a practice	○	○	○	○
C12. The requisites a team should prepare in order to successfully adopt a practice	○	○	○	○
C13. The cause of the problem team may encounter	○	○	○	○
C14. The solution a team may use to solve the problem	○	○	○	○

C15. The general knowledge based on experiences related to agile practice a team should learn	○	○	○	○

Please explain your answers to the above, concern by concern, if your answer is "Irrelevant"

_____

_____

_____

_____

_____

QII.1. To what extent do you agree that information provided by the tool related to each concern below is correct?

	Strongly disagree	Disagree	Somewhat agree	Agree	Strongly agree
C1. The objective/goal a team can achieve by adopting an agile practice	○	○	○	○	○
C2. The agile value a team can achieve by adopting a practice	○	○	○	○	○
C3. The agile principle a team can achieve by adopting a practice	○	○	○	○	○
C4. The activity a team should perform as part of a practice	○	○	○	○	○
C5. The problem a team may encounter while adopting a practice	○	○	○	○	○
C6. The situation of the team which is bad for adopting a practice	○	○	○	○	○
C7. The situation of the team which is good for adopting a practice	○	○	○	○	○
C8. The activity that they perform which is bad for adopting a practice	○	○	○	○	○
C9. The activity that they perform which is good for adopting a practice	○	○	○	○	○
C10. The artifact required for adopting a practice	○	○	○	○	○
C11. The role required for adopting a practice	○	○	○	○	○
C12. The requisites a team should prepare in order to successfully adopt a practice	○	○	○	○	○
C13. The cause of the problem team may encounter	○	○	○	○	○

C14. The solution a team may use to solve the problem	○	○	○	○	○
C15. The general knowledge based on experiences related to agile practice a team should learn	○	○	○	○	○

Please explain your answers to the above, concern by concern, if your answer is **disagree** or **strongly disagree**

_____

_____

_____

_____

_____

QII.2. To what extent do you agree that the amount of information, provided by the tool, related to each concern below is ***good enough*** to satisfy your needs?

	Strongly disagree	Disagree	Somewhat agree	Agree	Strongly agree
C1. The objective/goal a team can achieve by adopting an agile practice	○	○	○	○	○
C2. The agile value a team can achieve by adopting a practice	○	○	○	○	○
C3. The agile principle a team can achieve by adopting a practice	○	○	○	○	○
C4. The activity a team should perform as part of a practice	○	○	○	○	○
C5. The problem a team may encounter while adopting a practice	○	○	○	○	○
C6. The situation of the team which is bad for adopting a practice	○	○	○	○	○
C7. The situation of the team which is good for adopting a practice	○	○	○	○	○
C8. The activity that they perform which is bad for adopting a practice	○	○	○	○	○
C9. The activity that they perform which is good for adopting a practice	○	○	○	○	○
C10. The artifact required for adopting a practice	○	○	○	○	○
C11. The role required for adopting a practice	○	○	○	○	○
C12. The requisites a team should prepare in order to successfully adopt a practice	○	○	○	○	○
C13. The cause of the problem team may encounter	○	○	○	○	○

C14. The solution a team may use to solve the problem	○	○	○	○	○
C15. The general knowledge based on experiences related to agile practice a team should learn	○	○	○	○	○

Please explain your answers to the above, concern by concern, if your answer is **disagree** or **strongly disagree**

_____

_____

_____

_____

_____

QIII. To what extent do you agree that information related to each concern below helps you to decide whether or not a practice is suitable for your team?

	Strongly disagree	Disagree	Somewhat agree	Agree	Strongly agree
C1. The objective/goal a team can achieve by adopting an agile practice	○	○	○	○	○
C2. The agile value a team can achieve by adopting a practice	○	○	○	○	○
C3. The agile principle a team can achieve by adopting a practice	○	○	○	○	○
C4. The activity a team should perform as part of a practice	○	○	○	○	○
C5. The problem a team may encounter while adopting a practice	○	○	○	○	○
C6. The situation of the team which is bad for adopting a practice	○	○	○	○	○
C7. The situation of the team which is good for adopting a practice	○	○	○	○	○
C8. The activity that they perform which is bad for adopting a practice	○	○	○	○	○
C9. The activity that they perform which is good for adopting a practice	○	○	○	○	○
C10. The artifact required for adopting a practice	○	○	○	○	○
C11. The role required for adopting a practice	○	○	○	○	○
C12. The requisites a team should prepare in order to successfully adopt a practice	○	○	○	○	○
C13. The cause of the problem team may encounter	○	○	○	○	○
C14. The solution a team may use to solve the problem	○	○	○	○	○

C15. The general knowledge based on experiences related to agile practice a team should learn	○	○	○	○	○

Please explain your answers to the above, concern by concern, if your answer is **disagree** or **strongly disagree**

_____

_____

_____

_____

_____

QIV. To what extent do you agree with the statement below?

	Strong disagree	Disagree	Somewhat agree	Agree	Strongly agree
Q4.1 I was able to find the concern I want to know easily	○	○	○	○	○
Q4.2 I was able to understand the answer to each concern easily	○	○	○	○	○
Q4.3 I was able to describe my team situation using the tool easily	○	○	○	○	○
Q4.4 I was able to describe my goal in adopting agile using the tool easily	○	○	○	○	○
Q4.5 I was able to gain knowledge/experience quickly about agile practice adoption using this tool	○	○	○	○	○
Q4.6 I was able to gain knowledge/experience easily about agile practice adoption using this tool	○	○	○	○	○
Q4.7 The tool is easy to use	○	○	○	○	○
Q4.8 I am satisfied with the tool	○	○	○	○	○
Q4.9 This tool helps me decide whether or not an agile practice is suitable for my team	○	○	○	○	○

Q5.1 Any concern(s) related to agile practice adoption you would like to add? If yes, please state below with the reason it should be added?

Q5.2 Any of our concern(s) you would like to modify? If yes, please state below with the reason it should be modified.

Q5.3 Other feedback you would like to give

Would you agree to participate in the next round of the survey by doing online?

- ○ Yes, I would
- ○ No, I would not

If you would agree to re-participate, please enter email address.

_____

Appendix B

Ontology Model Validation: Survey Results

Following tables present the complete results of the four survey questions with the same data structure to Table 7.2.

Concern	Never	Rare	Sometimes	Often	Always	Median
C1	2	3	7	3	**8**	3
C2	5	5	**8**	3	2	3
C3	5	6	**9**	2	1	3
C4	4	2	4	**10**	3	4
C5	2	5	4	**6**	**6**	4
C6	3	2	6	4	**8**	4
C7	4	1	5	5	**8**	4
C8	4	4	5	**6**	4	3
C9	4	3	**7**	6	3	3
C10	5	4	3	**8**	3	4
C11	**6**	5	4	3	5	3
C12	4	5	**7**	3	4	3
C13	1	4	**7**	6	5	3
C14	2	5	4	**7**	5	4
C15	**6**	5	5	5	2	3

Table B.1 Results of Question 1.1. Consider your own experience with agile, how often do you need the information related to each concern before you start adopting each agile practice?

Concern	Irrelevant	Useful	Important	Necessary	Median
C1	1	4	8	**10**	3
C2	3	**10**	4	6	2
C3	3	**12**	1	7	2
C4	3	3	**9**	8	3
C5	1	7	7	**8**	3
C6	2	5	**11**	5	3
C7	2	7	**8**	6	3
C8	2	7	**8**	6	3
C9	2	7	**8**	6	3
C10	2	**8**	**8**	5	3
C11	4	6	**7**	6	3
C12	2	**7**	6	**7**	3
C13	2	**8**	6	**8**	3
C14	2	**9**	2	**10**	3
C15	2	**12**	4	5	2

Table B.2 Results of Question 1.2. How would you rate the relevancy level of each concern to the agile practice adoption?

Concern	S Disagree	Disagree	Sw Agree	Agree	S Agree	Median
C1	2	1	8	**10**	2	4
C2	0	3	**9**	**9**	2	3
C3	1	3	**9**	**9**	1	3
C4	0	2	7	**12**	2	4
C5	0	2	8	**9**	4	4
C6	1	1	**11**	8	2	3
C7	1	1	**12**	7	2	3
C8	1	1	**9**	**9**	3	4
C9	1	1	9	**10**	2	4
C10	0	2	8	**9**	4	4
C11	0	2	8	**10**	3	4
C12	0	1	9	**10**	3	4
C13	2	3	**8**	7	3	3
C14	1	4	7	**9**	2	3
C15	0	4	**9**	6	4	3

Table B.3 Results of Question 2.1. To what extent do you agree that information provided by the tool related to each concern is correct?

Concern	S Disagree	Disagree	Sw Agree	Agree	S Agree	Median
C1	2	3	**10**	7	1	3
C2	1	2	**9**	8	3	3
C3	1	3	**11**	7	1	3
C4	1	1	**11**	9	1	3
C5	1	2	**12**	6	2	3
C6	1	2	**10**	9	1	3
C7	1	2	**10**	9	1	3
C8	1	2	**10**	9	1	3
C9	1	2	**10**	**10**	0	3
C10	1	2	7	**10**	3	4
C11	1	1	**9**	**9**	3	4
C12	1	1	**10**	9	2	3
C13	2	6	**8**	7	0	3
C14	3	4	7	**9**	0	3
C15	1	3	**10**	7	2	3

Table B.4 Results of Question 2.2.To what extent do you agree that the amount of information, provided by the tool, related to each concern is good enough to satisfy your needs?

	Q4.2	**Q4.3**	**Q4.4**
P1	(1) Agile ways of working require first and foremost a change in mindset and in management culture. (2) Focus more on how to improve individual mindset, behaviour, interaction inside and outside of the team		
P2	Take into account the external parameters which are influencing the team behaviour	Take a bit of distance from the agile manifesto, which is a bit outdated.	Congrats. Huge job. I see the potential, providing that the underlying cause/effect correlations are taken to the next level.
P3	(1)Relationships and trust between people (2) Context is Key (3) Support of Leadership influences adoption (4) Change of habit that requires adapted mindset and mindset is non-observable, only behaviours.		Keep on experimenting ! Congrats for the efforts already achieved.

P4		It's not up to a tool to let me decide what practice is suitable to my team. The idea of agile is to check with the team how THEY would like to improve. What's the next step THEY are ready to experiment with.my primary concern with the tool is that I can't see how this tool fosters team experimentation in a safe way.	Thanks for your time... I'd encourage you to spend a few weeks with teams working on an agile journey to get a feeling of their real needs. I'm not sure this tool is the right answer to the biggest issue they face during agile adoption
P5			It might be a benefit for the tool to propose other kinds of views (hyperlinked, multi-concepts tables, ...) in order not to constraint the user in "one" unique mode for gathering information.
P6			As a researcher or a developer, we are not in charge of adopting a practice. So all the concerns are somehow irrelevant to us. What is important to us is why we need to practice it and what are the activities we should follow.

Table B.5 Result of Question.4

Appendix C

Supporting Tool

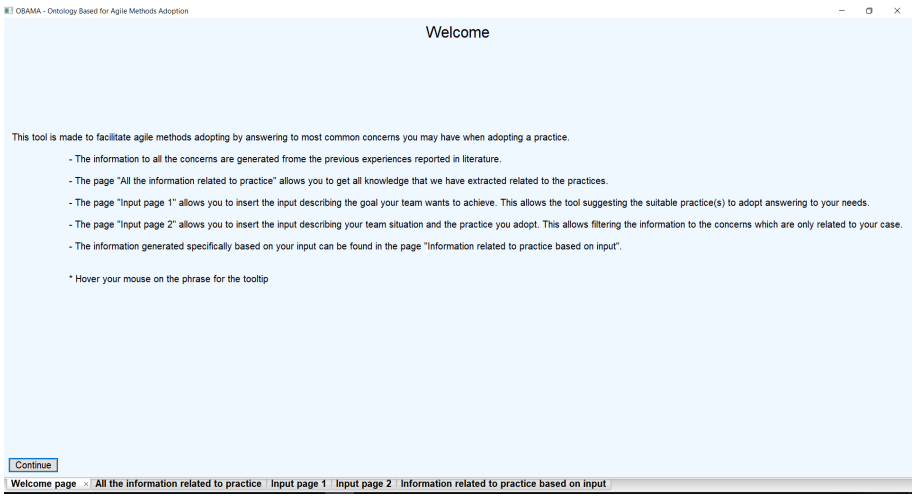


Fig. C.1 Welcome page

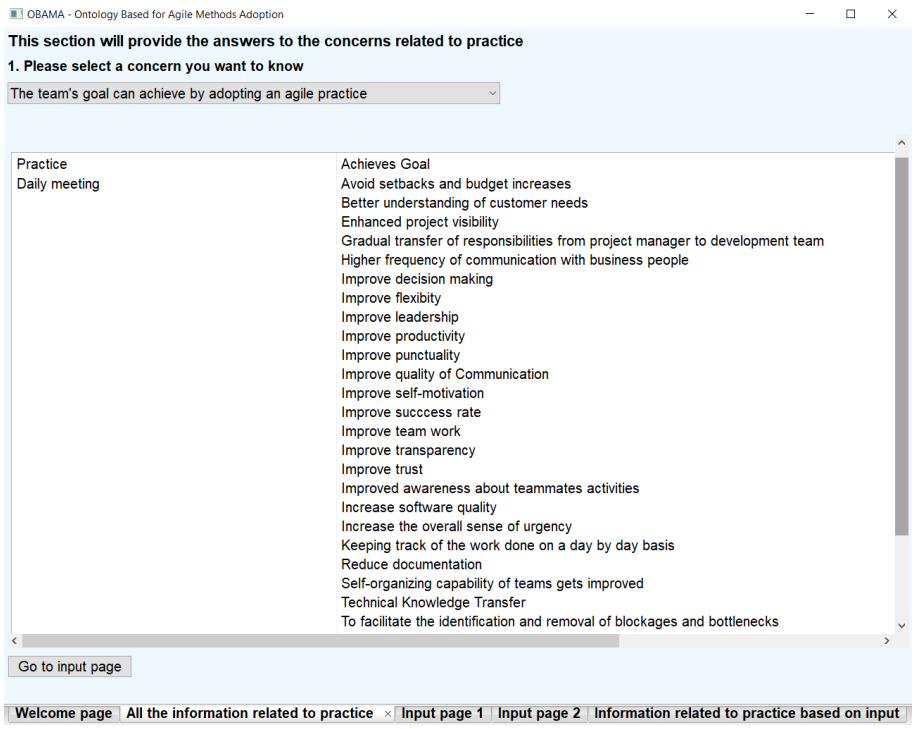


Fig. C.2 The goal a team can achieve by adopting an agile practice

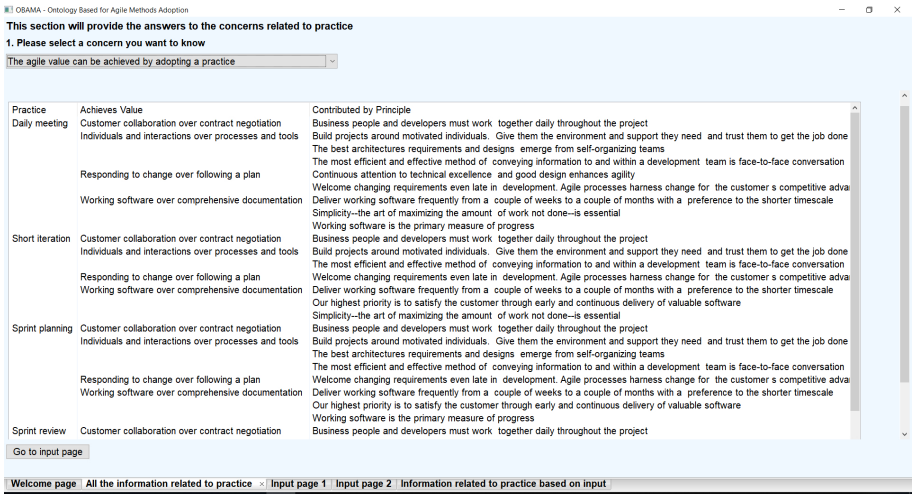


Fig. C.3 The agile value a team can achieve by adopting a practice

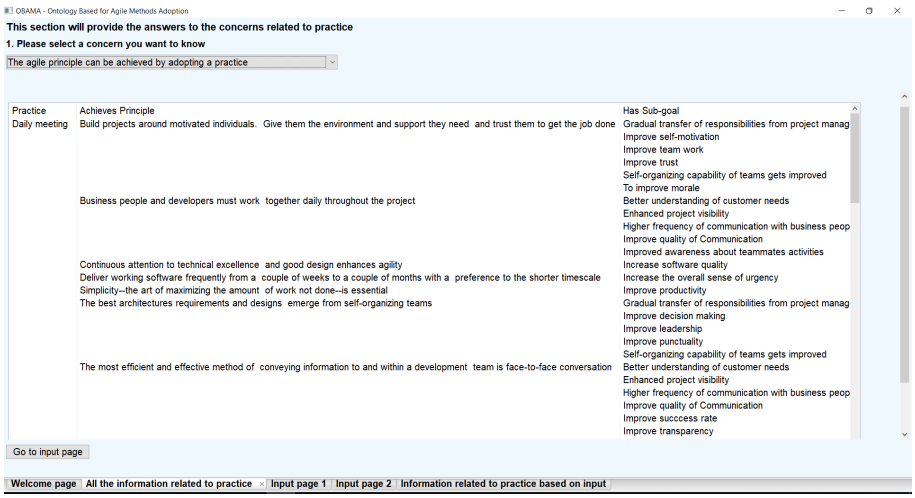


Fig. C.4 The agile principle a team can achieve by adopting a practice

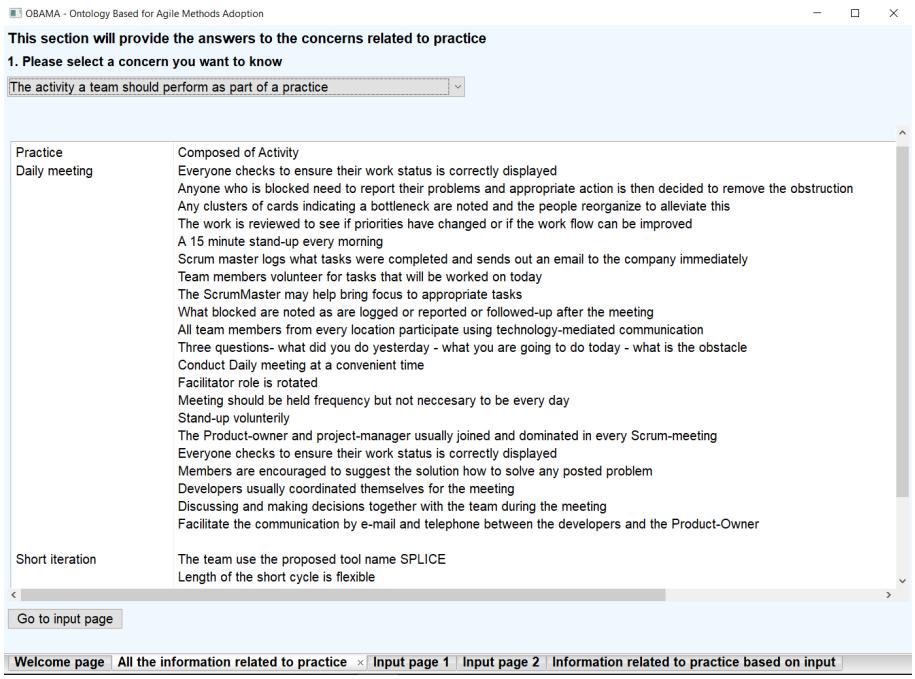


Fig. C.5 The activity a team should perform as part of a practice

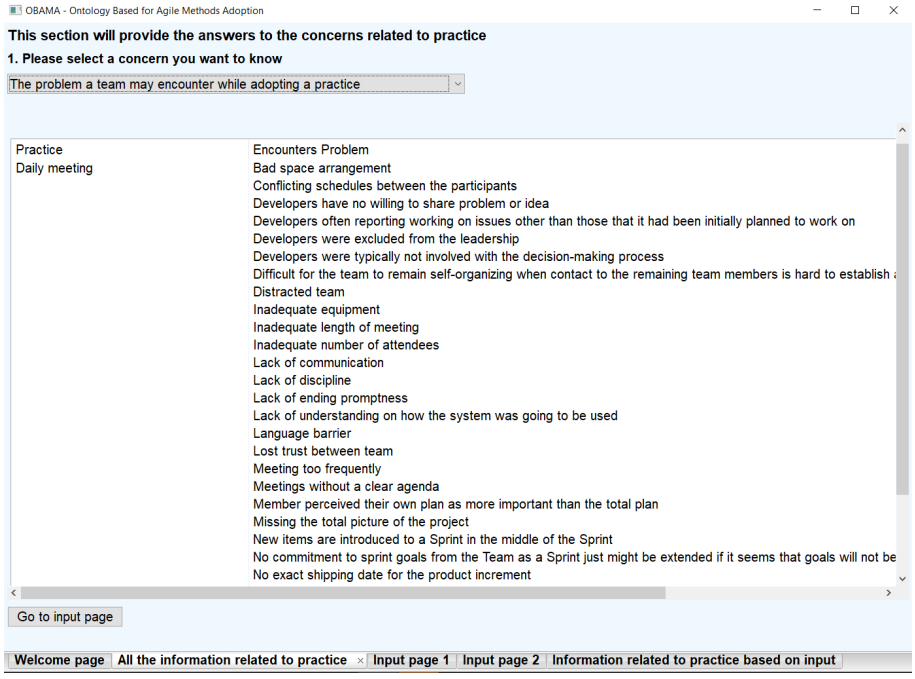


Fig. C.6 The problem a team may encounter while adopting a practice

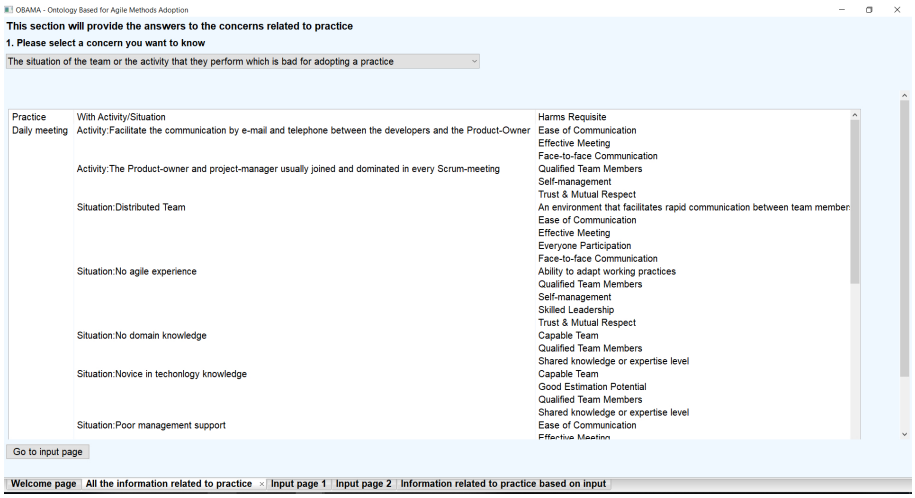


Fig. C.7 The situation of the team or the activity that they perform which is bad for adopting a practice

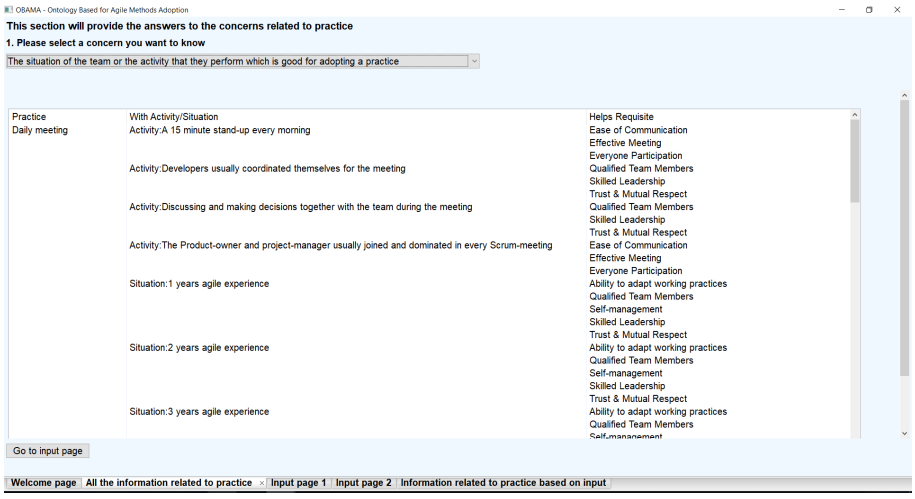


Fig. C.8 The situation of the team or the activity that they perform which is good for adopting a practice

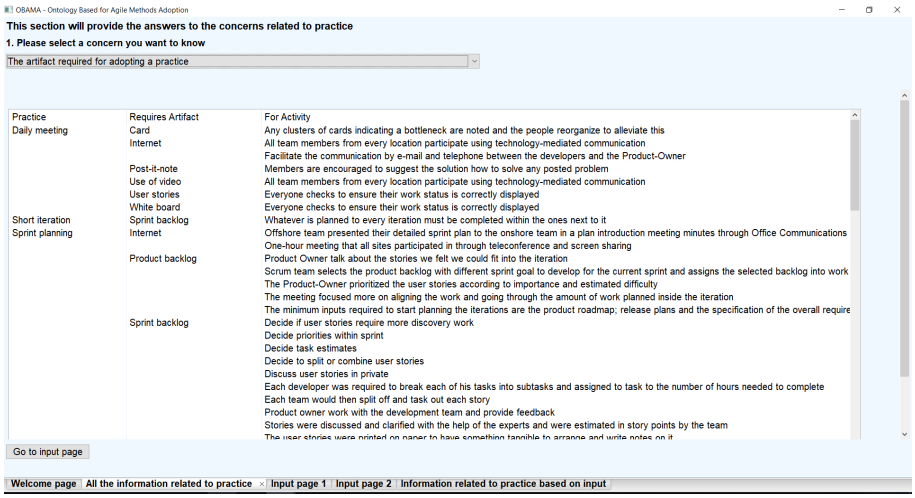


Fig. C.9 The artifact required for adopting a practice

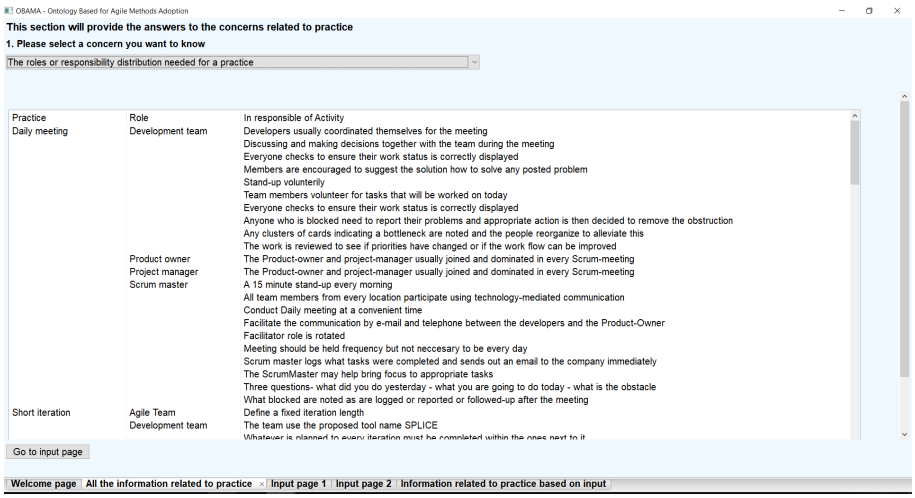


Fig. C.10 The role required for adopting a practice

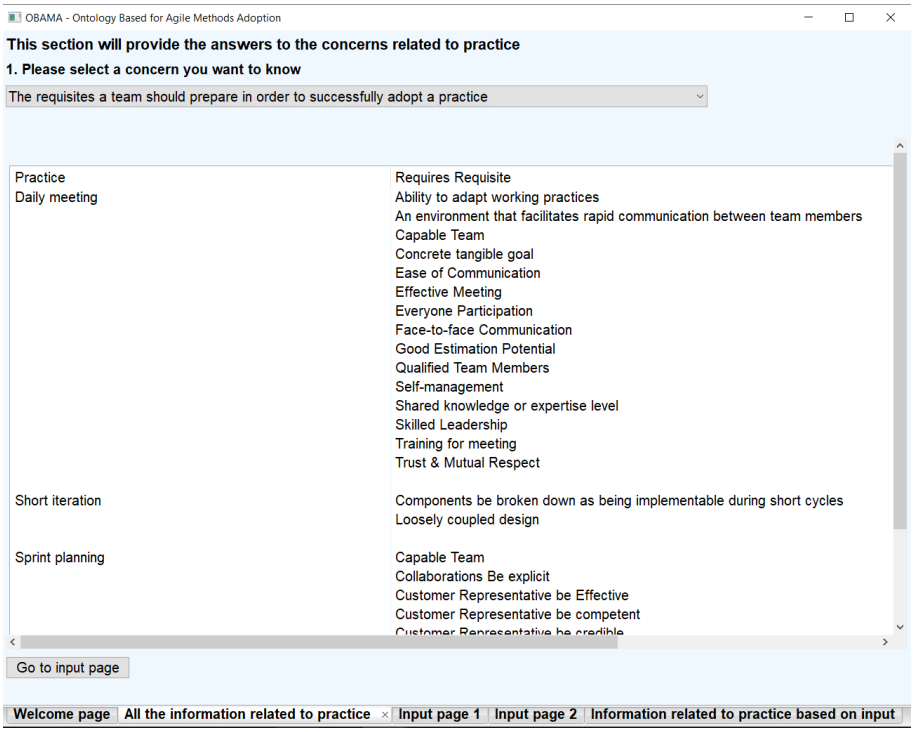


Fig. C.11 The requisites a team should prepare in order to successfully adopt a practice

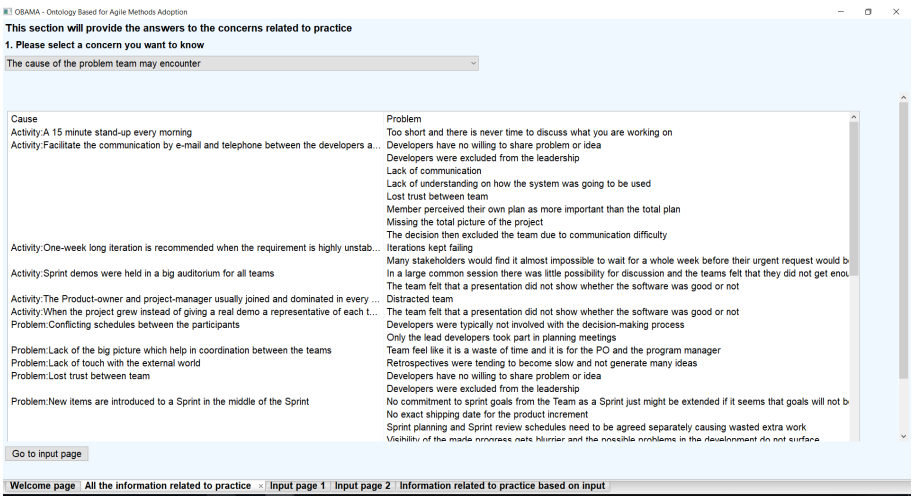


Fig. C.12 The cause of the problem team may encounter

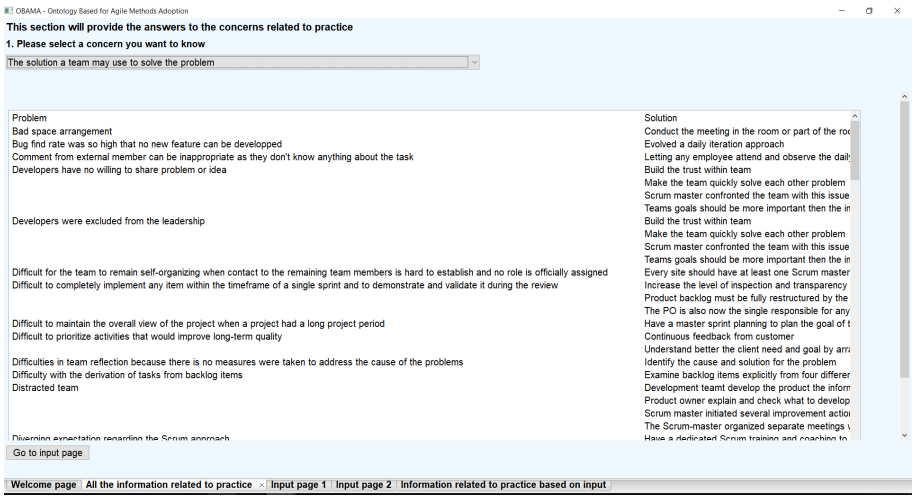


Fig. C.13 The solution a team may use to solve the problem

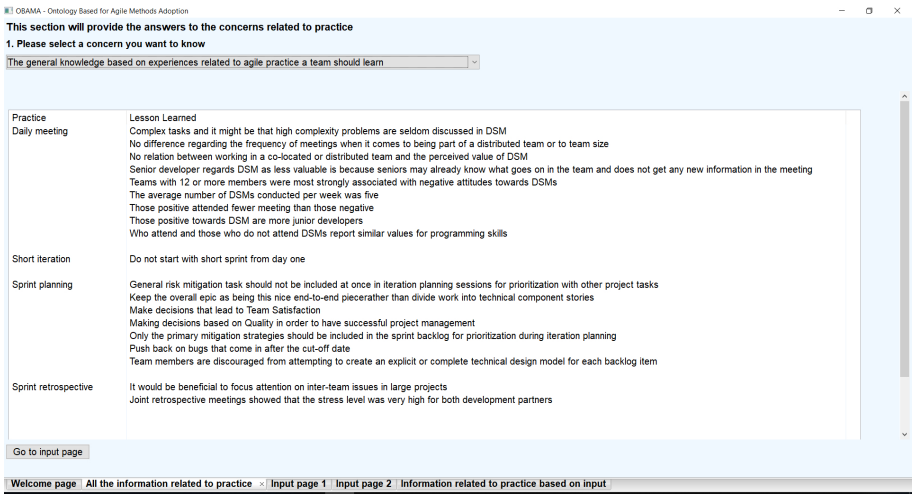


Fig. C.14 The general knowledge based on experiences related to agile practice a team should learn

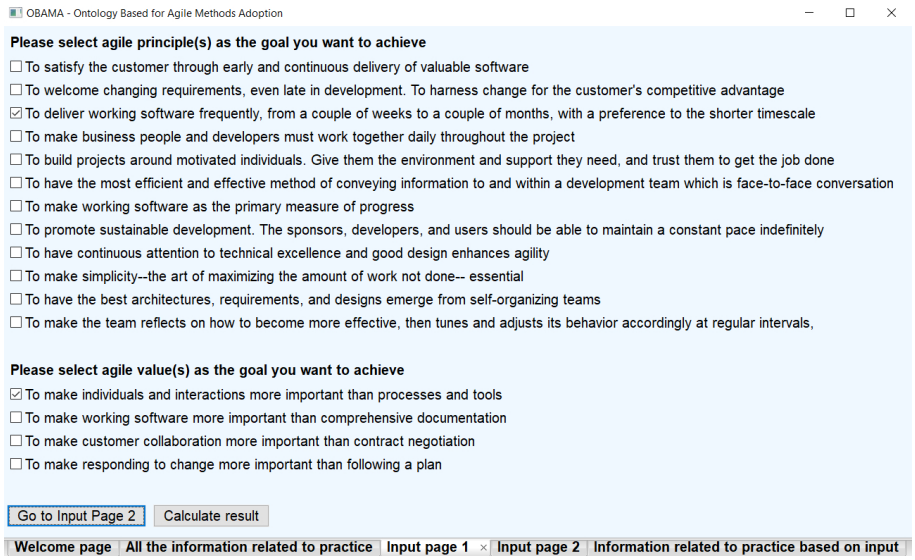


Fig. C.15 Input page 1- For selecting agile values and principles

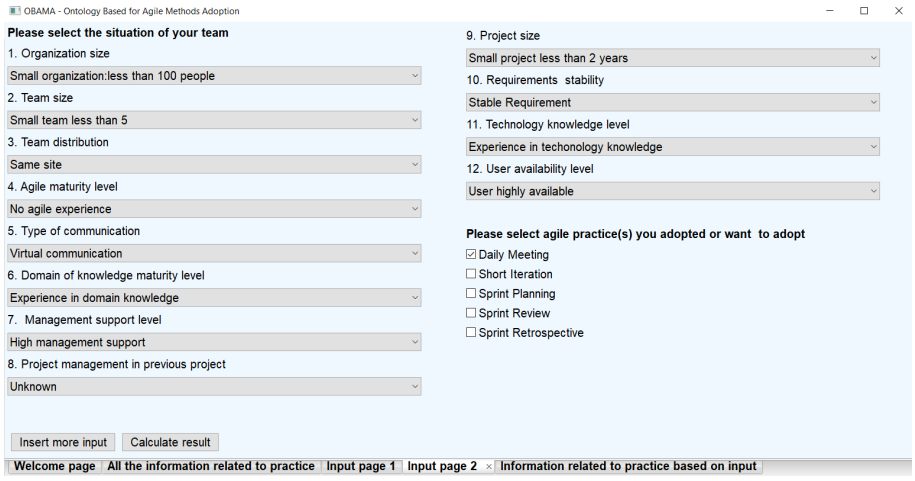


Fig. C.16 Input page 2 - For describing team's situations

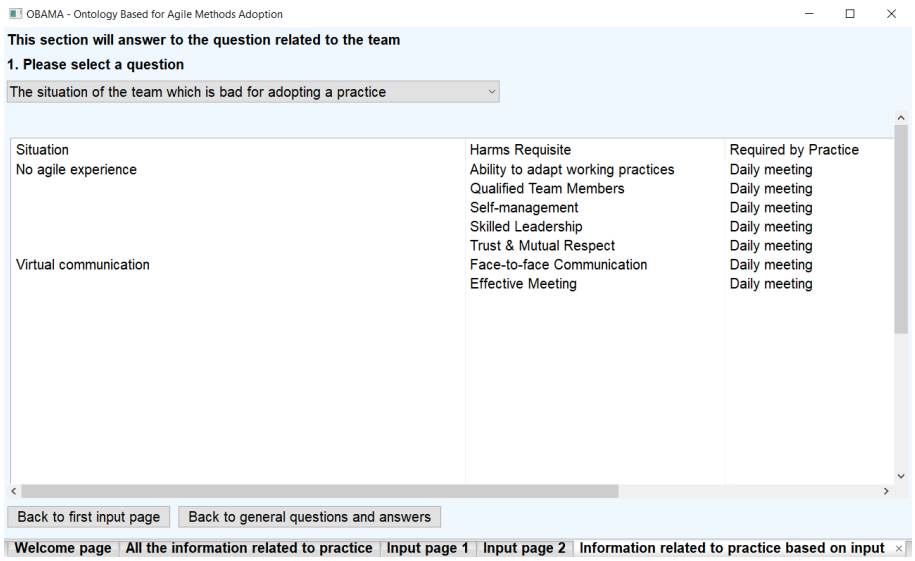


Fig. C.17 The situation of the team or the activity that they perform which is bad for adopting a practice based on inputs

Appendix D

Socio-intentional diagrams for agile methods tailoring

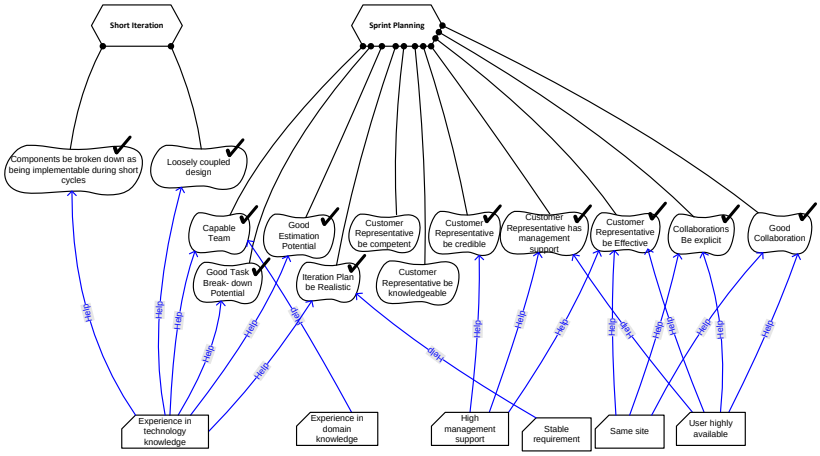


Fig. D.1 Relationship between Short Iteration and the requisites for their success and team's situation visualized in iStar 2.0

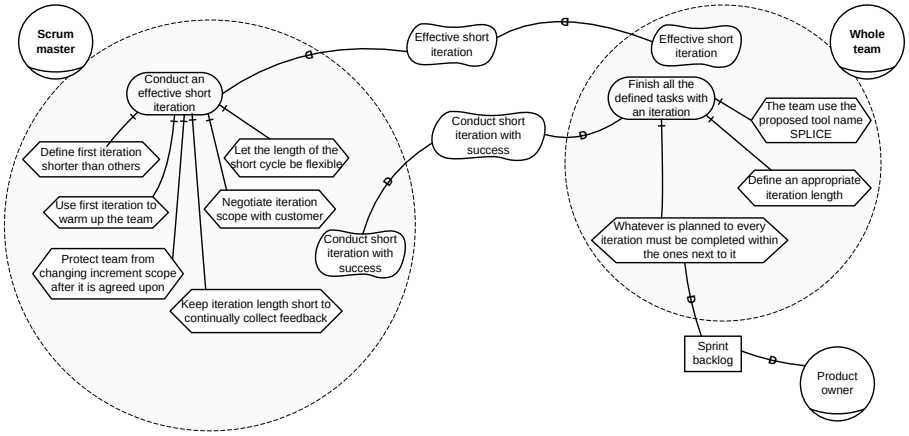


Fig. D.2 Activities of *Short iteration* visualized in Star 2.0

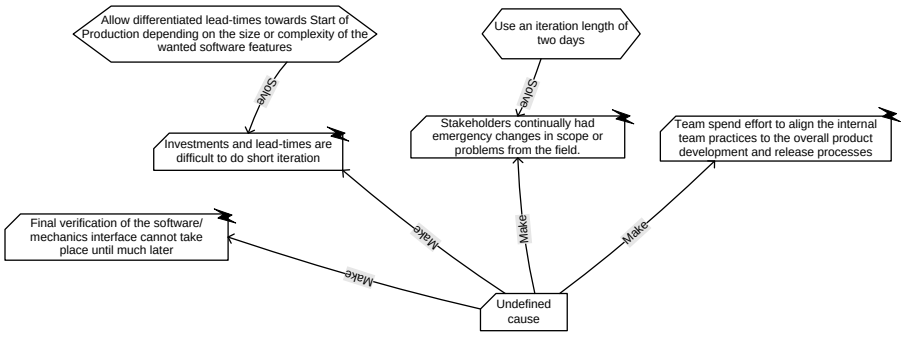


Fig. D.3 Cause, Problems in *Short iteration* and Solution visualized in iStar 2.0