An Intentional Perspective on Partial Agile Adoption

Soreangsey Kiv[1], Samedi Heng[1], Manuel Kolp[1] and Yves Wautelet[2]
[1]*LouRIM-CEMIS, Université Catholique de Louvain, Belgium*
[2]*Faculty of Economics and Business, KULeuven, Belgium*

Keywords: Agile Methods, Partial Adoption, Goal Modeling, Conceptual Model, Requirements Engineering.

Abstract: Nowadays, the agile paradigm is one of the most important approaches used for software development besides structured and traditional life cycles. To facilitate its adoption and minimize the risks, different meta-models have been proposed trying to unify it. Yet, very few of them have focused on one fundamental question: How to *partially* adopt agile methods? Intuitively, choosing which practices to adopt from agile methods should be made based on their most prioritized goals in the software development process. To answer this issue, this paper proposes a model for partial agile methods adoption based on intentional (i.e., goal) perspectives. Hence, adoption can be considered as defining the goals in the model, corresponding to the intentions of the software development team. Next, by mapping with our goal-based model, suitable practices for adoption could be easily found. Moreover, the relationship between roles and their dependencies to achieve a specific goal can also be visualized. This will help the software development team to easily identify the vulnerabilities associated with each goal and, in turn, help to minimize risks.

1 INTRODUCTION

In user-intensive software development, to effectively deal with quality expectations, change or risk management, and to ensure that all the requirements are satisfied, activities such as requirements gathering, design, development and testing should be iterative and incremental. In the last decade, among other approaches such as the introduction of eXtreme Proramming (XP) (Beck, 2000), Feature-Driven Development (FDD) (Palmer and Felsing, 2001), Dynamic Systems Development Method (DSDM) (Stapleton, 1997), Crystal family (Cockburn, 1998), Scrum (Schwaber and Beedle, 2002), etc., this led to the definition of the Agile Manifesto (Fowler and Highsmith, 2001; Abrahamsson et al., 2003) to offer alternatives to traditional approaches.

Among many research subjects in the *agile community*, partial agile adoption has gained a lot of interests. It has always been actively studied and known as agile tailoring, customization or practice selection, etc. (Campanelli and Parreiras, 2015). Based on the survey on partial agile adoption in the same publication indicated that 56 out of 783 papers published between 2002 and 2014 actually describe agile method tailoring approaches. The common objective is to offer the possibility to adopt agile methods gradually or partly, rather than fully. According to the Campanelli and Parreiras, by adopting agile partially, organizations can avoid the need of spending efforts and resources on irrelevant practices.

Intuitively, when software teams want to partially adopt agile methods – either one or the combination of multiple methodologies – they should have in mind the reasons they want to do it and which are the goals they want to achieve after the adoption. (Tripp and Armstrong, 2014) conduct an exploratory study to find out that different motives exist for agile adoption, all with different project management configurations focusing on agile practices for software development. (Campanelli and Parreiras, 2015) highlight that 42% of the papers working on agile tailoring use goal as one of their criterion for practice selection. For instance, among others, the Strategic Analysis for Agile Practices framework (Esfahani et al., 2011) proposes to link the selection of agile practices to the organization's business goals.

Unfortunately, these business goals along with other criteria are often described textually for partial agile adoption. To ensure wide diffusion in academia, we believe that explicit formalisms should have been proposed by using meta-models (Henderson-Sellers and Gonzalez-Perez, 2005; Mikulėnas et al., 2011; Esfahani et al., 2010) to describe the methods. This has motivated us to propose a partial agile adoption from a goal perspective by means of a concep-

Kiv, S., Heng, S., Kolp, M. and Wautelet, Y.
An Intentional Perspective on Partial Agile Adoption.
DOI: 10.5220/0006429301160127
In *Proceedings of the 12th International Conference on Software Technologies (ICSOFT 2017)*, pages 116-127
ISBN: 978-989-758-262-2

tual model to offer a more explicit representation and guideline. Our motivation contrasts with previous goal-based agile tailoring approaches in the sense that we employ the original goal behind agile methods creation (i.e., the Agile Manifesto) rather than business goal. This perspective coincides also with the work of (Madi et al., 2011) which emphasizes that agile practices should follow the agile values which is fundamental to define the culture of the software company.

To summarize, we aim here at building a generic goal-based conceptual model to facilitate partial agile method adoption. Each agile concept will be seen as a goal to achieve, to which dependencies between roles and resources are associated. A complementary objective is to explain how a software team can use our model at the two levels of adoption process: *tactical* and *operational*.

- At the tactical level, the software team has to choose the practices to integrate into its existing development process.
- At the operational level, agile practices will be implemented.

This paper is organized as follows. Section 2 discusses various agile methods and meta-models proposed in literature. Their pros and cons are highlighted. Section 3 presents our framework for adopting agile methods partially. We first expose our agile conceptual model adopting tactical and operational levels. Then, we discuss on the use of the intentional i* framework to instantiate our conceptual model. Last, we provide a methodology for using our framework in selecting agile practices. Section 4 provides a validation for our approach. Finally, we conclude the paper in Section 5 including a discussion on future directions.

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 an objective to put more focus on project management organization while XP is designed to be more responsive to customer requirement changes (Mikulėnas et al., 2011). The research from (Jacobson et al., 2006) shows that agile methods tend to be compatible with small development teams and focus more on people rather than processes or artifacts. Other well-known predecessors of agile frameworks can be found in the literature, even inspired from other disciplines than software and information systems engineering: Agile Unified Process (AUP) (Ambler, 2005), OpenUP (Kroll and MacIsaac, 2006) and Agile Modeling (Ambler, 2002), Kanban (Ahmad et al., 2013; Liker, 2004), Lean Software Development (Poppendieck and Poppendieck, 2003), Lean Office and Organization (Chiarini, 2013), etc.

These agile methods are generally described textually as a series of values, principles and practices (Beck, 2000; Schwaber and Beedle, 2002; Fowler and Highsmith, 2001). To formalize their description, help their adoption, minimize the risks of deploying them and generalize them, meta-models (Henderson-Sellers and Gonzalez-Perez, 2005; Mikulėnas et al., 2011; Esfahani et al., 2010) have been created. Meta-models can then help the adoption of such methods better, faster and at the same time minimizing the risks. They aim at making relations, attributes, control flows, rules of a particular process more appealing based on a particular perspective, e.g., products and capability assessment (Henderson-Sellers and Gonzalez-Perez, 2005), partial agile adoption (Mikulėnas et al., 2011), and goal-oriented (Esfahani et al., 2010).

(Schwaber, 2004) has designed a meta-model to support the construction of agile methods. By relying on the measurement concept from the Situational Method Engineering framework (Henderson-Sellers et al., 2014), it provides guidance to agile methodologists during the construction and throughout the development process itself. Rather than working on a single agile method, (Mikulėnas et al., 2011), on their side, focus on a meta-model enabling a fusion of different agile methods – named 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. This offers great flexibility in adopting agile methods since one can manually select and combine different alternative components coming from different methods at will.

Interestingly, (Lin et al., 2014) and (Esfahani et al., 2010) have shed more light on goal-oriented meta-models. (Lin et al., 2014) propose a novel goal-oriented method on the top of the AUP (Ambler, 2005), called GOAUP that could be also applied to OpenUP and other adaptation of the *Unified Process* to agile. Goal-Net theory (Shen et al., 2004) is used to model the hierarchical goals in the AUP using a Goal-Net diagram. Each iteration can be considered to have one main goal. (Esfahani et al., 2010) propose a more sophisticated goal-oriented meta-model that put focus on social interaction, in which human relations are clearly defined. For example, the relationship/dependency between roles in required processes and the skills are associated with each goal in turn

described with certain vulnerabilities to minimize the risks of the adoption process. Although many other meta-models have been proposed (Lin et al., 2014; Shen et al., 2004; Mikulėnas et al., 2011; Esfahani et al., 2010; Bézivin, 2004; Schuppenies and Steinhauer, 2002; Damiani et al., 2007; Henderson-Sellers and Gonzalez-Perez, 2005), none of them focuses specifically on partial agile adoption in an intentional goal-oriented perspective. The closest research related to our work is (Esfahani et al., 2010), in which goal-oriented has been used to model micro-processes (i.e., practices). However, the authors focus rather on how to visualize agile methods, mainly Scrum, in social modeling while our work aims at using a goal-oriented framework specifically dedicated to partial agile methods adoption.

3 PROPOSED FRAMEWORK

Our objective is to introduce a generic conceptual model based on goal and social aspects to describe agile software development. To do this, we formulate our proposed framework in two main parts.

First, we create a generic conceptual model, applicable to any agile method. We will need to define a common structure that can, theoretically, represent all the agile elements from a goal perspective. Each agile concept will be seen as a goal to achieve, to which dependencies between roles and resources are associated.

Second, we propose making use of the intentional i* framework (Yu et al., 2011) (that means distributed intentionality) as schema instances derived from the conceptual model to visualize all the dependencies/relationships. The choice is inspired by the fact that i* framework has stimulated a considerable interest in a socially-motivated approach to systems modeling and design, and has led to a number of extensions and adaptations (Eric, 2009). We will explain how this goal-based conceptual model and its i* representation can offer advantages at both tactical and operational levels for a partial agile methods adoption process.

We describe, hereafter, more formally our conceptual model and the mapping of each component into i* elements. We then apply it on two agile methods, Scrum and XP to verify that all the agile concepts can be mapped to our conceptual model and to show that it is helpful for practitioner in selecting practices and identifying vulnerability.

3.1 Agile Conceptual Model

Our conceptual model is created to help the software team to select a set of practices for adoption based on goals. The Agile Manifesto introduces four values, twelves principles and the abstraction of practice (Fowler and Highsmith, 2001) to agile methods should respect. Agile values are the large-scale criteria we use to judge what we see, what we think and what we do (Madeyski, 2010). They are fundamental where a set of practices can be followed on their basis (Madi et al., 2011). According to agile practitioners, knowing the most important agile values is the key to follow the best set of practices. Practices are concrete activities and practical techniques used to develop and manage software projects in accordance with the agile principles (Sidky et al., 2007). There is a big gap between values and practices, since values are too abstract to directly guide the development. Therefore, principles come into agile and act as a bridge to narrow this (Madeyski, 2010). We introduce the concept of *agile feature* based on (Laanti et al., 2013) as the composition of principles. As the concept of principle, it is introduced to further narrow the gap between principle and practice. Finally, a practice is performed by a certain role (actor) with/without using/producing the artifact. Our conceptual model is built on top of these abstractions: *value*, *principle*, *agile feature*, *practice*, *role* and *artifact*. Figure 1 is the conceptual model of agile methods in the goal perspective.

The definition of the main components in our conceptual model are:

- *Value* is the large-scale criteria used to judge what we see, what we think and what we do (Madeyski, 2010). It is seen as the top-level goal in agile methods adoption.
- *Principle* helps to establish a mind-set for solid software engineering practices (Pressman, 2005). It comes into agile methods and acts as a bridge to narrow the gap between values and practices (Madeyski, 2010).
- *Agile feature* is the distinctive characteristic of each principle, as defined by (Laanti et al., 2013). These features can help to narrow the gap between principles and practices, and consequently the relevant practices can be better identified by the development team.
- *Practice* is the concrete activity and practical technique that is used to develop and manage software projects in a manner consistent with the agile principles (Sidky et al., 2007).
- *Role* in agile methods refers to the allocation of specific roles through which the software produc-

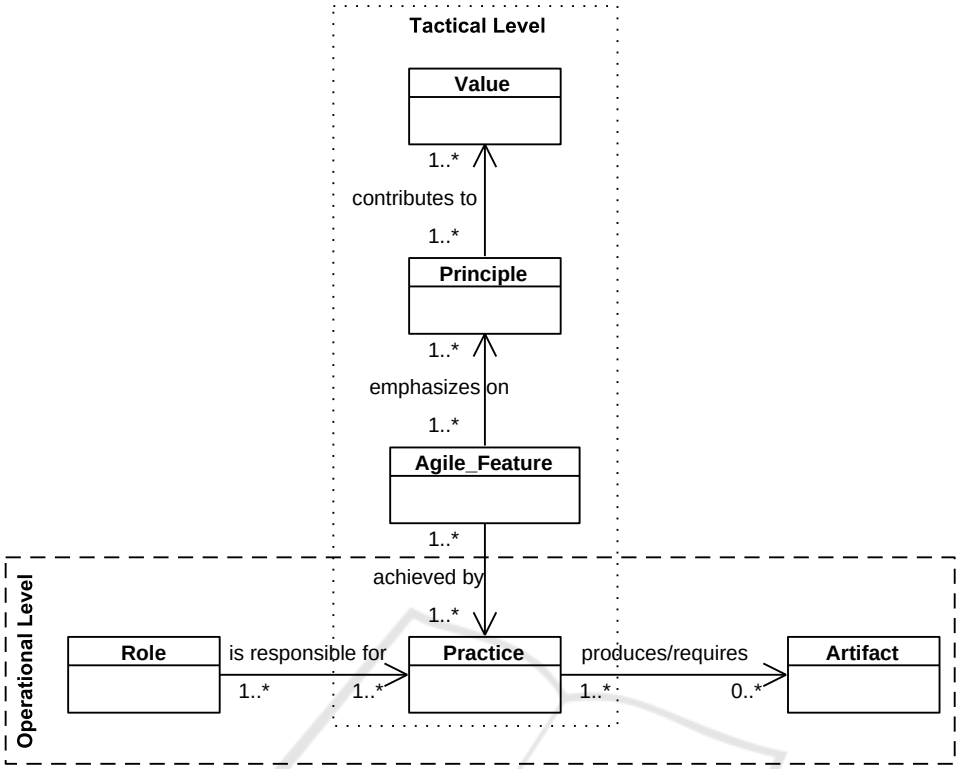

Figure 1: Agile conceptual model.

tion in a development team is carried out (Sidky et al., 2007).

- *Artifact* is the resource produced during software development. It can be tangible like user stories (Cohn, 2004; Wautelet et al., 2014) or intangible like working software functionality.

All the relationships between different components are of the type "many-to-many".

3.2 Mapping the Agile Conceptual Model to i*

Goal/Intention and social dependencies between each component over a specific goal can be visualized using goal-oriented frameworks. In this research, we use the i* framework (Yu et al., 2011). It has two main models: the Strategic Dependency model (SD) and Strategic Rationale model (SR). The SD model describes external dependency relationships between goals, actors using hard-goal, soft-goal, task and resource elements. These four kinds of dependency are suitable to represent respectively the functional goal, functional goal without a clear-cut of satisfiability, functional goal with a specific activity or practice to achieve and resource or artifact needed during the development. In addition to the SD, the SR model provides a deeper representation to the internal, intentional dependency with the means-end, task-decomposition and contribution links to describe stakeholders' interests and the (combination of) activities, sub-goals, artifacts that help to achieve the main goal. A full description of i* can be found in (Yu et al., 2011) but we summarize the important concepts below for self-sufficiency:

- *Hard-goal* is an intentional desire of an actor, the specific way of how the goal is to be satisfied is not described.
- *Soft-goal* is similar to (hard) goals except that the criteria for the goals satisfaction are not clear-cut, it is judged to be sufficiently satisfied from the point of view of the actor.
- *Task* is a particular way of attaining a goal.
- *Resource* is the finished product of some deliberation-action process.
- *Role* is an abstract characterization of the behavior of a social actor within some specialized context or domain of endeavor. Its characteristics are easily transferable to other social actors.
- *Contribution link (some+)* is a positive contribution whose strength is unknown.

Table 1: Mapping agile concepts to i*.

Agile concepts	**i* concepts**
Principle	Soft-goal
Agile Feature	Soft-goal
Practice	Task
Artifact	Resource
Role	Role
Contributes to	Contribution link (some+)
Emphasizes on	Contribution link (and or make)
Achieved by	Contribution link
Is responsible for	Task dependency
Produces/requires	Resource dependency

- *Contribution link (and)* is kind of contribution where the parent is satisfied if all of the offsprings are satisfied.
- *Contribution link (make)* is a positive contribution strong enough to satisfy a soft-goal.
- *Task dependency* is a relationship where the depender depends on the dependee to carry out an activity (task).
- *Resource dependency* is a relationship where the depender depends on the dependee for the availability of an entity (physical or informational).

In order to visualize our model using the i* framework, all the concepts and their relationships must be mapped to i* elements. This mapping is performed with an heuristic search using the idea of Cartesian product. Indeed, for every concept in our model, we compare its definition/objective to all the possible i* elements that can be found. The best match should have the most similar objective/definition. We first map the classes and we finish by mapping their relationships. We managed to map all the concepts and their relationships to i* elements and detail hereafter the motivation of our mapping for each concept:

- *Value*, *principle* and *agile feature* are represented as soft-goals. These concepts are described in agile methods as the objectives to be achieved without any defined criterion. For instance, "Individuals and interactions over processes and tools" explains why software team members and their interactions are important for the development but there is no explicit explanation of how the team should be, what kind of interactions they need in order to attain the benefit of this value. Similar explanations apply for the concepts of principle and agile feature.
- *Practice* is described as a set of activities that the software team must follow in order to realize its benefit. It is represented as a task in i*.
- *Artifact* is the resource required or created when performing a practice. For example "user stories" are the resources needed for a "sprint planning" and from this practice, a "sprint backlog" is created. In i*, artifact is represented as resource.
- *Role* exists in many forms in different agile methods. Each role has different responsibility and can be played by different or the same actor. It is represented as a role in the i* framework.
- *Contributes to* is the relationship between value and principle which are soft-goals. It is represented as a contribution link with the "some+" tag, as the strength of its contribution to satisfy is unknown.
- *Emphasizes on* is the relationship between principle and agile feature. Each principle is decomposed into different parts used to emphasize it. This kind of relationship is represented by a contribution link with the tag "and" while principle is the parent and can be fully satisfied only if all of the children, agile features, are satisfied. Some principles having only one feature, the "contribution link" with the tag "make" is used to represent them. That one feature is the only thing to be satisfied to fulfill the principle it emphasizes.
- *Achieved by* is the relationship between agile feature (soft-goal) and practice (task). This relationship is a "contribution link" where the tag used depends on the practice to carry out and the feature to achieve.
- *Is responsible for* is the relationship between an agile role and the practice under its responsibility. It is represented by task dependency.
- *Produces/requires* is the relationship between an

agile role and the artifact it requires or creates. It is represented by an i* resource dependency, which roles depend on each other for the resource needed.

3.3 Agile Practices Selection Process

The partially adopt agile methods in our framework consists of four steps and follows an iterative and incremental process as represented by Figure 2.

- *Defining Goals*: within this stage, agile practitioners need to define the goals/objectives they want to achieve after the agile methods adoption, either single or multiple methods.
- *Selecting Practices*: based on the goals defined previously, the software team follows our goal-based conceptual model to find out the supporting practices. At this point, the team should be able to select which practices to adopt.
- *Checking Vulnerabilities*: in order to help the team to identify the possible risks during practices implementation, various relationships/dependencies between agile practices, roles, and resources needed will be visualized to the organization. This enables the software team to see the possible vulnerabilities, e.g., roles or resources required can not be provided.
- *Solving Vulnerabilities or Implementing Practices*: finally, based on the results of dependency check, if there is any identified risks associated with practice, the software team needs to make the decision on how to avoid them: either by solving the cause of vulnerability or by redefining the goal (eventually, the process of selecting practices is started again). If there is not any risk, the team can start implementing the practices into their development process.

4 VALIDATION

In order to fully validate the framework, there are two necessary validations should be carried out, we would call *theoretical* and *practical* validation. *Theoretical* validation aims at verifying that all the agile concepts can always be mapped to our conceptual model and showing that it can help practitioner in selecting practices and identifying vulnerability. The *practical* validation aims at validating whether or not the framework is useful for the development team in the real case of partial agile methods adoption. However, as the preliminary study, we only focus on theoretical validation in this paper. As we can not take practices from all the agile methods to discuss in one paper, we choose to work only with the two most popular agile methods namely Scrum and XP.

The validation will be done in two steps. At the first step, the "tactical level", the relationships between *value*, *principle*, *agile feature* and *practice* will be visualized. Practitioners can follow our methodology to find out what are the practices that fulfill their desired goals. Then, during selected practices implementation, named "operational level", the dependencies between practices, roles and resources needed will be visualized. This level is supposed to help practitioners to identify the vulnerability.

We describe below these two levels applied to Scrum and XP.

4.1 Application at Tactical Level

Our goal-based conceptual model can first be used at "the tactical level" for attempting to partially adopt agile methods.

One of the most common questions in partial agile adoption is: *which practices should we adopt first?* To answer this, the software team can visualize the adoption process as having a set of goals to achieve and follow the conceptual model until reaching the lowest level in order to find out the practices they need to achieve the goals.

Over the years, many values, principles and practices have been proposed in different agile methods. However, to avoid an overwhelming amount of information, this research will discuss only on the root four values, twelve principles and some practices proposed in the two most popular agile methods: Scrum and XP. It is important to remind that the main idea of this paper is not to map all the existing elements, but rather to show the interests of having them mapped in a goal perspective.

We can discover which principles contribute to which value, which practices can help to achieve which principle, etc. by either a literature review or a qualitative research approach. In our case, the first method has been carried out while the second one will be done later.

Although there is no explicit claim of relationship between them, they can be understood from the textual meaning. For example, "Individuals and interactions over processes and tools" can easily be understood as the ability of the software team and the interactions that contribute to this value. Logically, all the principles that contribute to this value are those focusing on the improvement of the software team and their interactions. Similarly, we can group the twelve principles into the four values defined in the Agile Mani-

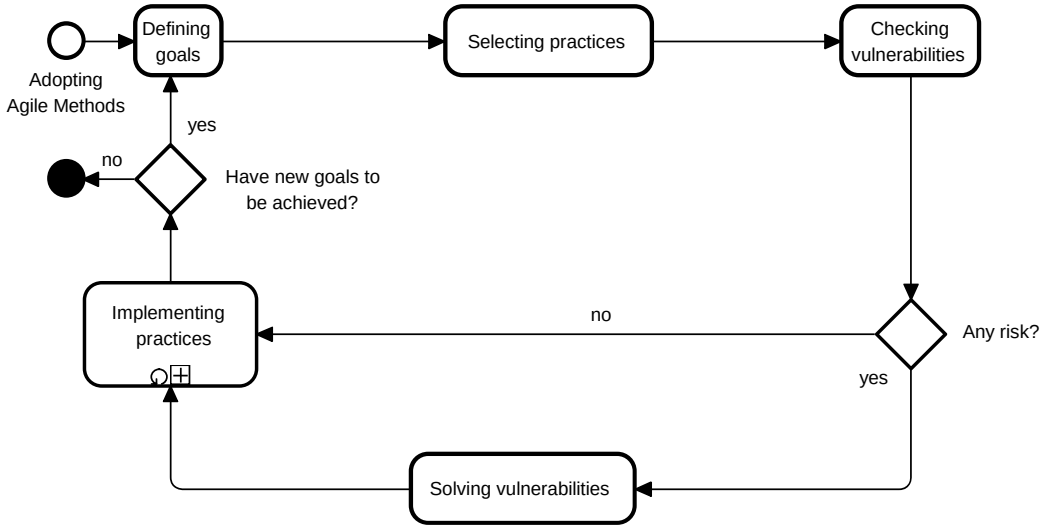

Figure 2: Agile practices selection process.

festo based on what they contribute to. Interestingly, (Laanti et al., 2013) analyze each principle in their research and introduces emphasis that we call "agile features".

Based on the knowledge extracted from (Martin, 2003; Laanti et al., 2013), we summarize the mapping between *value*, *principle* and *agile feature* in Table 2.

As mentioned earlier, it is not possible in this research to extract all the practices from all methods adopted by the industry. The two most popular methods have then been studied. Besides, being the most favored, Scrum and XP are known to be compatible and have frequently been adopted together (VersionOne, 2016).

- *Scrum*, the most used among agile methods (VersionOne, 2016) is a framework within which people can address flexible adaptive problems, while productively and creatively delivering products of the highest possible value (Schwaber and Sutherland, 2011). It is not a process or a technique for building products; rather, it is a method within which we can employ various processes and techniques. It consists of Scrum Teams and their associated roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to the success and usage of Scrum. The rules of Scrum bind together the events, roles, and artifacts, governing the relationships and interaction between them. Scrum has never been described with clear-cut practices. However, the list of practices can be found at (AgileAlliance, 2017).
- *XP*, as described in (Beck, 2000), is a process for the business of software development that make the whole software team focus on common, reachable goals. Using XP values and principles, software teams apply appropriate practices in their own context. XP practices are chosen for their encouragement of human creativity and their acceptance of human frailty. One of the goals of XP is to bring accountability and transparency to software development, to run software development like any other business activity. Another goal is to achieve more effective and efficient development with far fewer defects than is currently expected. Finally, XP aims to achieve these goals by celebrating and serving the human needs of everyone touched by software developmentsponsors, managers, testers, users, and programmers.

A list of XP and Scrum practices can be found on (AgileAlliance, 2017). The mapping between agile features and practices in this research has been created based on the understanding of each practice and its benefit described in (Martin, 2003; Schwaber and Beedle, 2002). The result is summarized in the online Appendix goo.gl/xg6aK4.

Figure 3 depicts the relationship between the different agile concepts: *value*, *principle*, *agile feature* and *practice*, represented in four different levels while the upper levels are contributed by the lower ones. Due to the limited space, only the practices that help to achieve the two principles contributing to a value are shown. In the Figure 3, the value "Individual and interaction over processes and tools" is contributed by two principles: "Build project around motivated individual. Give them the environment and support they need and trust them to get the job done" and "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation". Each principles is empha-

Table 2: Mapping between agile values, principles and features.

Value	Principle	Agile feature
V1: Individuals and interactions over processes and tools	**P1:** Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	**F1:** Motivated individuals **F2:** Good environment **F3:** Support **F4:** Trust
	P2: The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	**F5:** Efficiency (for conveying information) **F6:** Communication
	P3: Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	**F7:** Sustainability **F8:** People
	P4: The best architectures, requirements, and designs emerge from self-organizing teams	**F9:** Self-organization
	P5: At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	**F10:** Built-in improvement of efficiency and behavior
V2: Working software over comprehensive documentation	**P6:** Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	**F11:** Customer satisfaction **F12:** Continuous delivery **F13:** Value **F14:** Early delivery
	P7: Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	**F15:** Frequent deliveries
	P8: Working software is the primary measure of progress.	**F16:** Measure progress via deliverables
	P9: Simplicity the art of maximizing the amount of work not done is essential	**F17:** Simplicity **F18:** Optimize work
V3: Customer collaboration over contract negotiation	**P10:** Business people and developers must work together daily throughout the project	**F19:** Collaboration
V4: Responding to change over following a plan	**P11:** Welcome changing requirements, even late in development.	**F20:** Adaptability **F21:** Competitiveness **F22:** Customer benefit
	P12: Continuous attention to technical excellence and good design enhances agility	**F23:** Focus on technical excellence

sized by multiple agile features such as “Trust”, “Support”, “Good environment” and “Motivated individuals”. Each agile feature can be achieved by multiple agile practices of the method Scrum and XP. For example, we can achieve the feature “Trust” by using the practices “Collective ownership” and “Sign up”.

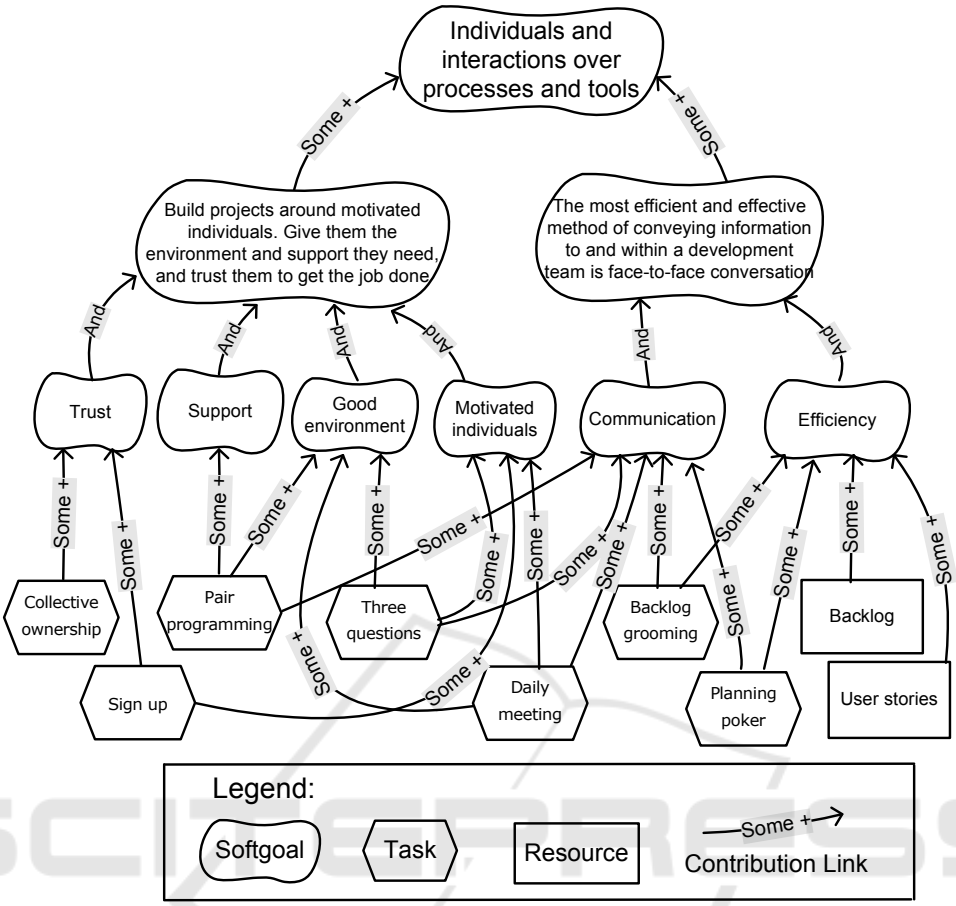

Figure 3: Relationship between value, principles, features and practices.

4.2 Application at Operational Level

The previous section describes how our conceptual model is created and can be used for partial agile adoption at the tactical level. We present here the operational level describing the importance of social interaction, particularly dependencies between roles, during agile practices implementation.

In the conceptual model, a role is in charge of carrying out a practice. By visualizing this concept in i*, one can easily understand how each role is involved in a particular process to achieve a goal. Dependencies between roles, goal and resources needed are then explicitly represented. The description of all the dependencies, critical roles and artifacts can help the organization and the software team to better analyze the chances of success or the probability of failure and thus be prepared to mitigate the risks. For instance, if many critical goals heavily depend on a particular role, the probability of failure increases when that role is not reliable. Thus, software teams must carefully assign it or adopt better practices to reduce or avoid such risks. Our conceptual model built so far contains only abstract definitions of practices. We show below that it can be instantiated on difference agile methods, as here in example, on Scrum and XP. Practices in each method are collected and mapped into agile features and the mapping list can be found in the Table 2. Figure 4 shows the dependencies between roles to perform the practices illustrated in Figure 3 at the operational level. Such representation is useful to show how different roles depend on each other to perform a certain tasks which can lead to achieve a specific goal at the tactical level. For instance, the Scrum master depends on the product owner to perform the following tasks: “backlog grooming”, “creating user stories”, “building the backlog” and “planning poker”. These practices are necessary in order to achieve the “efficiency” agile feature. If the product owner fails to perform them, this goal will most probably also fail. To avoid that, it is necessary to make sure that the scrum master manages to interact with the product owner despite tight schedules or to choose other practices which do not depend on the product owner.

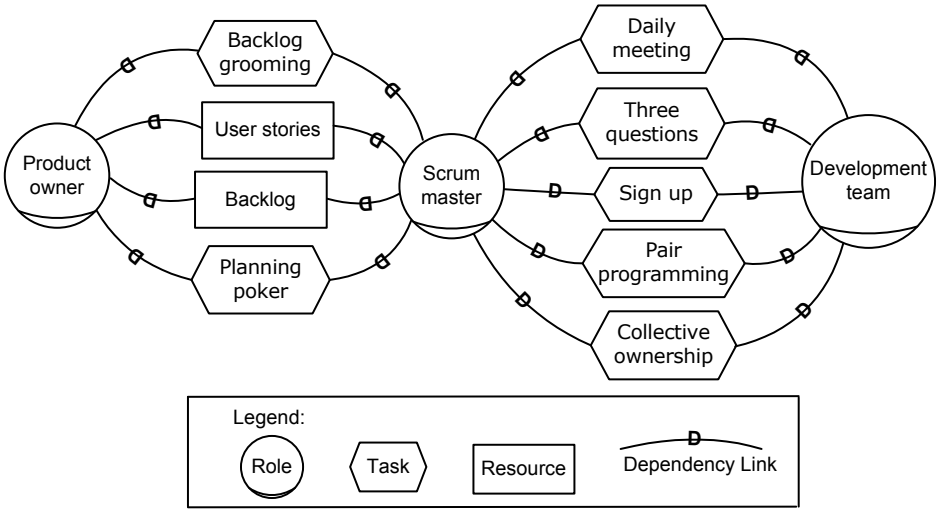

Figure 4: Scrum methodology at the operational level.

When practices in different methods are merged, so will do the roles. The software team has to decide which roles to keep and which can be eliminated. In our example, there are only two practices from XP in which the XP coach depends on the developer to perform "pair programming" and "collective ownership". Since the XP coach is just a depender which is not crucial, it can be replaced by the Scrum master. As for the developer role, even if it is necessary to perform the practices, it is already included in the software team.

5 CONCLUSION

This paper introduced a new conceptual model to represent agile methods in a goal perspective and allows partial method adoption. Our model offers two main advantages. First, the goal perspective enables the software team to quickly understand the objective of each concept (i.e., value, principle,agile feature and practice) and consequently enables the software team to select only the relevant concepts for partial agile adoption. Second, using the i* framework to visualize the relationships between roles and their intentional dependencies allows the team to identify the possible vulnerabilities resulting from adopting the chosen practices. Hence, the software team can avoid risks of project failure by reinforcing the role performance or choosing the alternative practices which can achieve the same goal.

While this conceptual model offers a lot of advantages, we acknowledge that a formal methodology is needed on top of it to practically validate it. Two possibilities can be chosen based on either a systematic review and a survey. We chose to focus on the empirical survey because we believe that it would offer a more evidential information from a practical perspective. At the time of writing, the survey is being conducted and we hope to publish the results in a next iteration of this research work.

To make sure that the software team can successfully adopt agile methods, we would also like to add another dimension for evaluating the specific process of adoption. In this regards, we aim not only at offering a roadmap for the software team to adopt agile methods, but also a set of performance indicators during the adoption process.

As our work aims at creating a generic conceptual model for any agile methods, it would be interesting if we can choose to adopt practices coming from different methods. This will enable organizations to have more variety of choices in term of practices. However, this should be done with caution. Ideally, to allow such possibility, one can introduce a weighting scheme, between different alternatives, as a risk indicator when adopting a particular practice. Finally, in the long term, we aim at developing a tool for showcasing our conceptual model.

REFERENCES

Abrahamsson, P., Warsta, J., Siponen, M. T., and Ronkainen, J. (2003). New directions on agile methods: A comparative analysis. In Clarke, L. A., Dillon, L., and Tichy, W. F., editors, *Proceedings of the 25th International Conference on Software Engineering, May 3-10, 2003, Portland, Oregon, USA*, pages 244–254. IEEE Computer Society.

AgileAlliance (2017). Subway map to agile practices, https://www.agilealliance.org/.

Ahmad, M. O., Markkula, J., and Oivo, M. (2013). Kanban in software development: A systematic literature review. In *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, pages 9–16. IEEE.

Ambler, S. (2002). *Agile modeling: effective practices for extreme programming and the unified process*. John Wiley & Sons.

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

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

Bézivin, J. (2004). In search of a basic principle for model driven engineering. *Novatica Journal, Special Issue*, 5(2):21–24.

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

Chiarini, A. (2013). *Lean Organization: from the Tools of the Toyota Production System to Lean Office*. Springer-Verlag.

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

Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

Damiani, E., Colombo, A., Frati, F., and Bellettini, C. (2007). A metamodel for modeling and measuring scrum development process. In Concas, G., Damiani, E., Scotto, M., and Succi, G., editors, *Agile Processes in Software Engineering and Extreme Programming, 8th International Conference, XP 2007, Como, Italy, June 18-22, 2007, Proceedings*, volume 4536 of *Lecture Notes in Computer Science*, pages 74–83. Springer.

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

Esfahani, H. C., Cabot, J., and Yu, E. S. K. (2010). Adopting agile methods: Can goal-oriented social modeling help? In Loucopoulos, P. and Cavarero, J., editors, *Proceedings of the Fourth IEEE International Conference on Research Challenges in Information Science, RCIS 2010, Nice, France, May 19-21, 2010*, pages 223–234. IEEE.

Esfahani, H. C., Yu, E. S. K., and Annosi, M. C. (2011). Towards the strategic analysis of agile practices. In Nurcan, S., editor, *Proceedings of the CAiSE Forum 2011, London, UK, June 22-24, 2011*, volume 734 of *CEUR Workshop Proceedings*, pages 155–162. CEUR-WS.org.

Fowler, M. and Highsmith, J. (2001). The agile manifesto. *Software Development*, 9(8):28–35.

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.

Henderson-Sellers, B., Ralyté, J., Ågerfalk, P. J., and Rossi, M. (2014). *Situational Method Engineering*. Springer.

Jacobson, I., Ng, P. W., and Spence, I. (2006). The essential unified process-a fresh start for process. *Dr Dobbs Journal*, 31(9):40–+.

Kolp, M., Giorgini, P., and Mylopoulos, J. (2002). Organizational multi-agent architectures: a mobile robot example. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 94–95. ACM.

Kolp, M. and Mylopoulos, J. (2001). Software architectures as organizational structures. In *Proc. ASERC Workshop on The Role of Software Architectures in the Construction, Evolution, and Reuse of Software Systems, Edmonton, Canada*.

Kroll, P. and MacIsaac, B. (2006). *Agility and Discipline Made Easy: Practices from OpenUP and RUP (Addison-Wesley Object Technology (Paperback))*. Addison-Wesley Professional.

Laanti, M., Similä, J., and Abrahamsson, P. (2013). Definitions of agile software development and agility. In McCaffery, F., O'Connor, R. V., and Messnarz, R., editors, *Systems, Software and Services Process Improvement - 20th European Conference, EuroSPI 2013, Dundalk, Ireland, June 25-27, 2013. Proceedings*, volume 364 of *Communications in Computer and Information Science*, pages 247–258. Springer.

Liker, J. K. (2004). *Toyota*. Esensi.

Lin, J., Yu, H., Shen, Z., and Miao, C. (2014). Using goal net to model user stories in agile software development. In *15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2014, Las Vegas, NV, USA, June 30 - July 2, 2014*, pages 1–6. IEEE Computer Society.

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

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.

Martin, R. C. (2003). *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

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.

Mylopoulos, J., Kolp, M., and Giorgini, P. (2002). Agent-oriented software development. In *Hellenic Conference on Artificial Intelligence*, pages 3–17. Springer Berlin Heidelberg.

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

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

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

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

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

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

Schwaber, K. and Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, 21.

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.

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.

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

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.

VersionOne (2016). 10th annual state of agile development survey.

Wautelet, Y., Heng, S., Kolp, M., and Mirbel, I. (2014). Unifying and extending user story models. In Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., and Horkoff, J., editors, *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings*, volume 8484 of *Lecture Notes in Computer Science*, pages 211–225. Springer.

Yu, E., Giorgini, P., Maiden, N., and Mylopoulos, J. (2011). *Social modeling for requirements engineering*. MIT Press.