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.

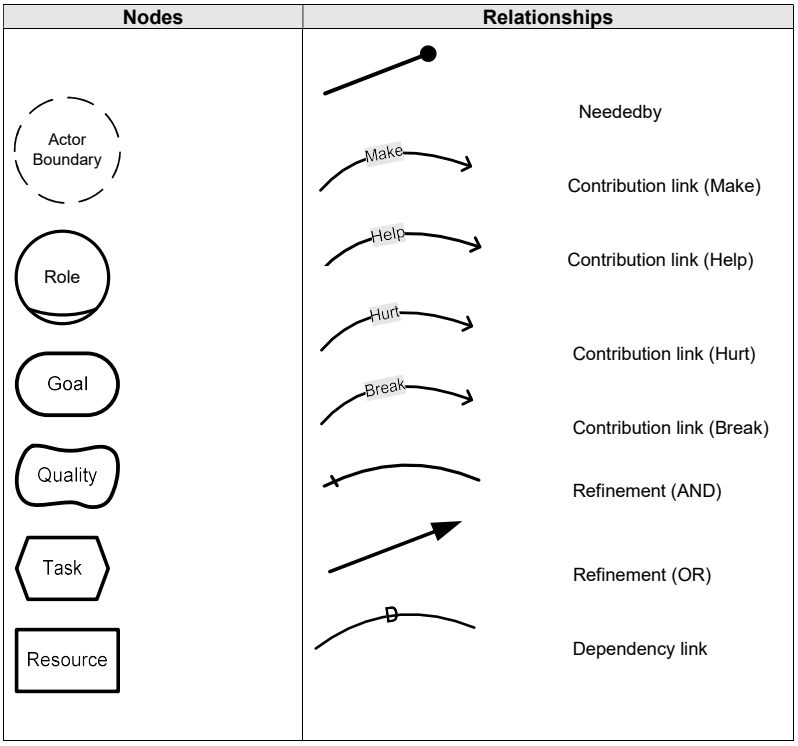


Fig. 8.1 iStar 2.0 notions

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.

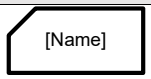
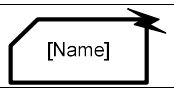
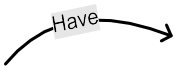
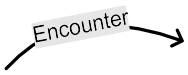
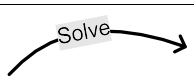
Agile element	Extended Notion	Definition
Situation		A state or situation that can affect (good or bad) another situation, goal or quality.
Problem		A bad state or situation that is conflict with another situation, goal or quality.
Have		A relationship “Have” between role and other elements
Encounter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

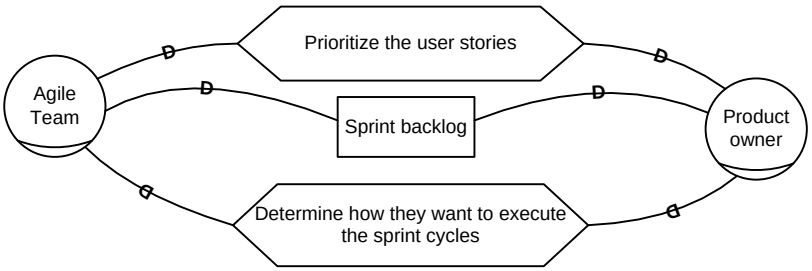


Fig. 8.3 Example of *Sprint planning* in SD view

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.

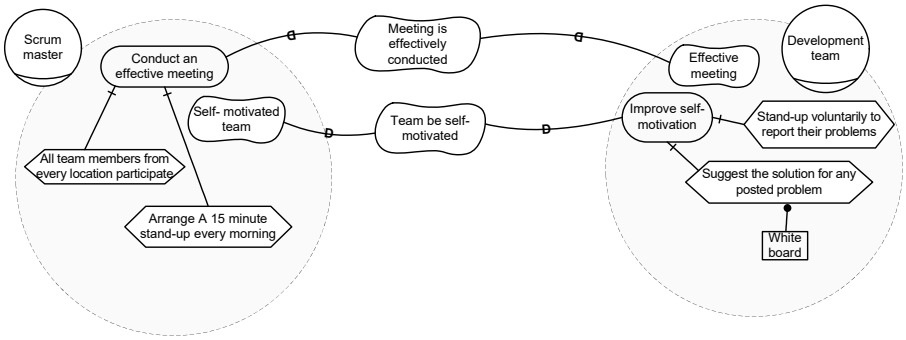


Fig. 8.4 Example of *Daily meeting* in SR view

8.2.2.3 NFR Framework

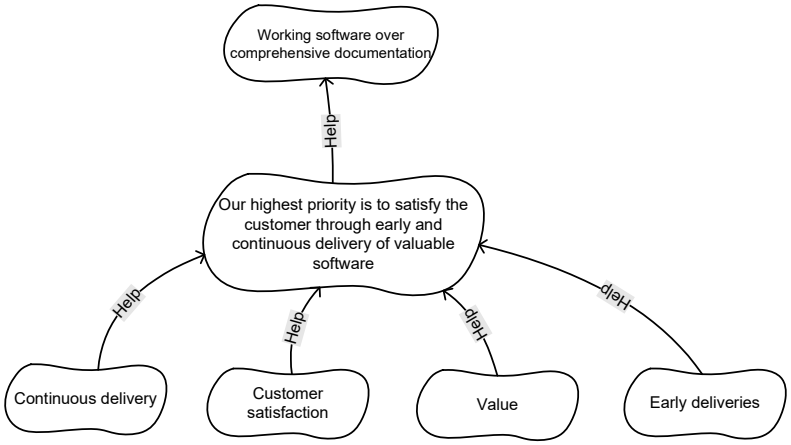


Fig. 8.5 Example of the relationship between agile value, principle, and goal in NFR Framework

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.

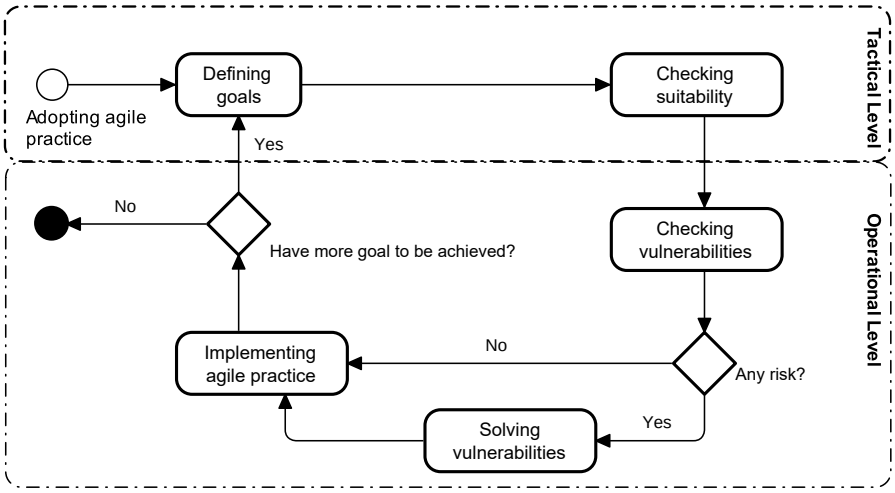


Fig. 8.6 Goal-oriented tailoring agile adoption process.

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.

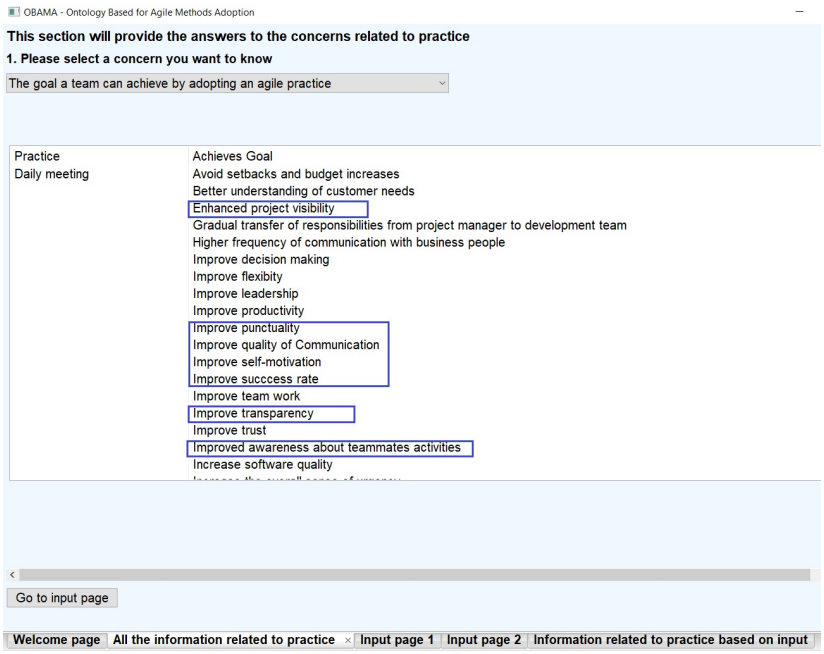


Fig. 8.7 Relationship between goals and agile practices listed by OBAMA-Tool

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

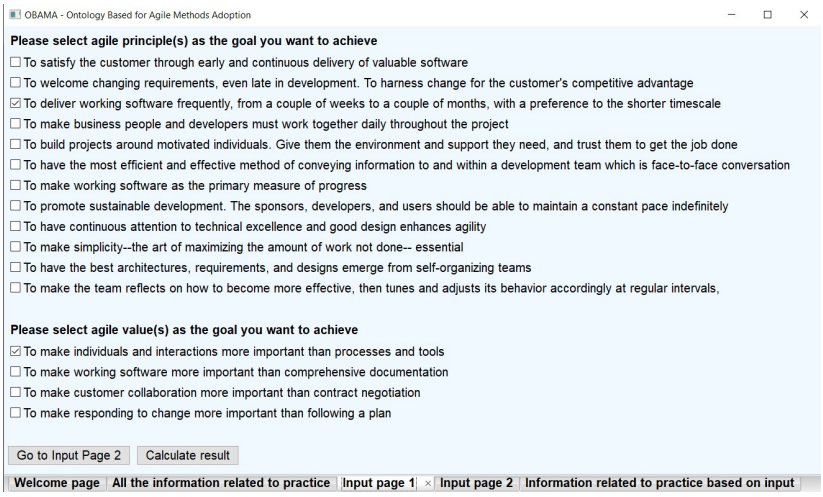


Fig. 8.8 Input page 1 of OBAMA-Tool for agile values and Principle

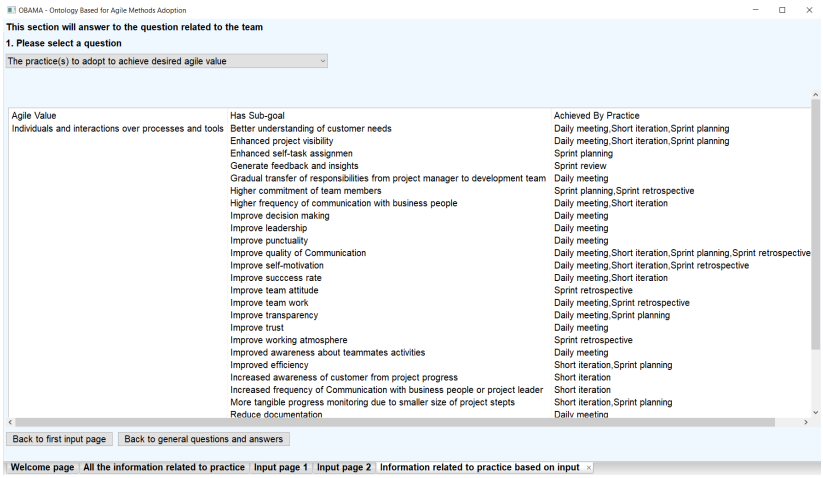


Fig. 8.9 Practices suggested by OBAMA-Tool to achieve selected Value

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.

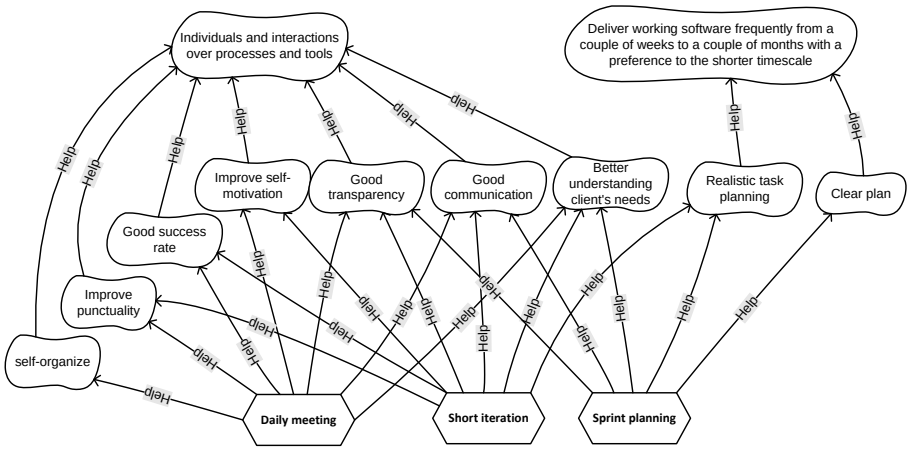


Fig. 8.10 Relationship between agile values, principles, goals and agile practices represented in iStar 2.0

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.

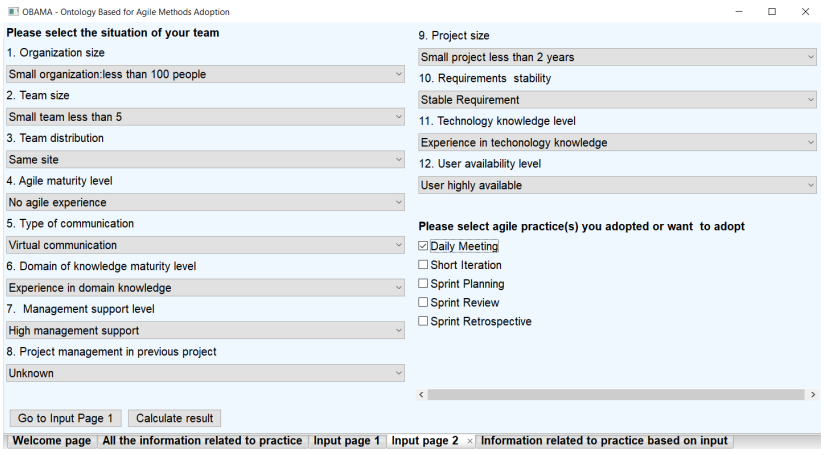


Fig. 8.11 Describing team's situations by using *Input page 2* of OBAMA-Tool

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*

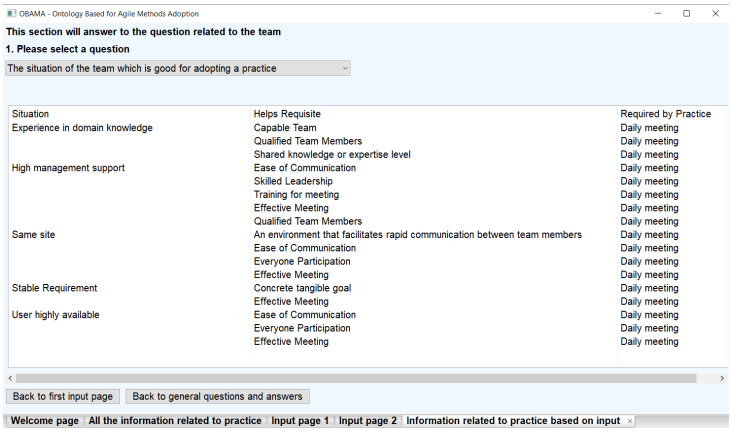


Fig. 8.12 Result of team's situations which is good for practice listed by OBAMA-Tool

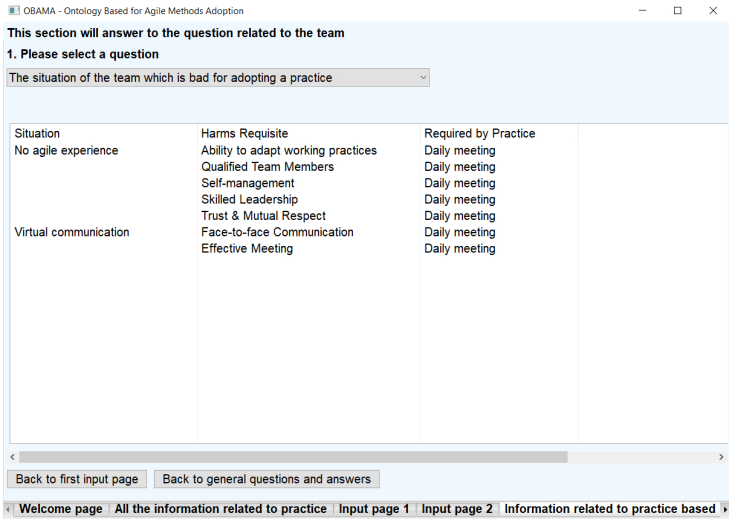


Fig. 8.13 Result of team's situations which is bad for practice listed by OBAMA-Tool

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.

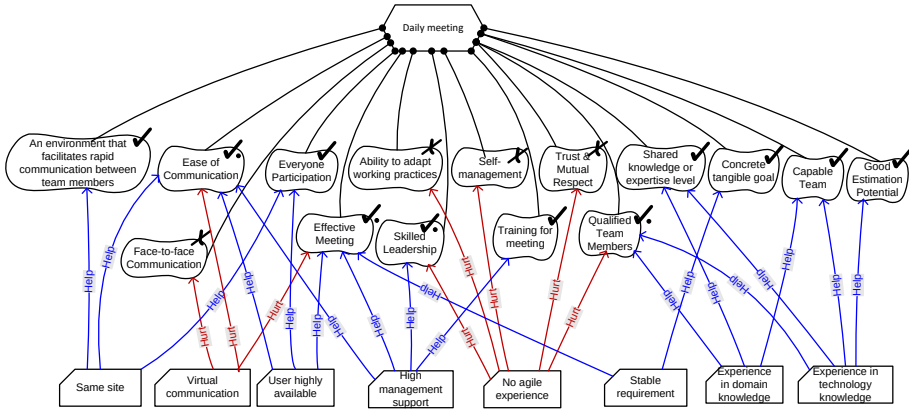


Fig. 8.14 Relationship between Daily meeting, the requisites for its success and team's situation visualized in iStar 2.0

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.

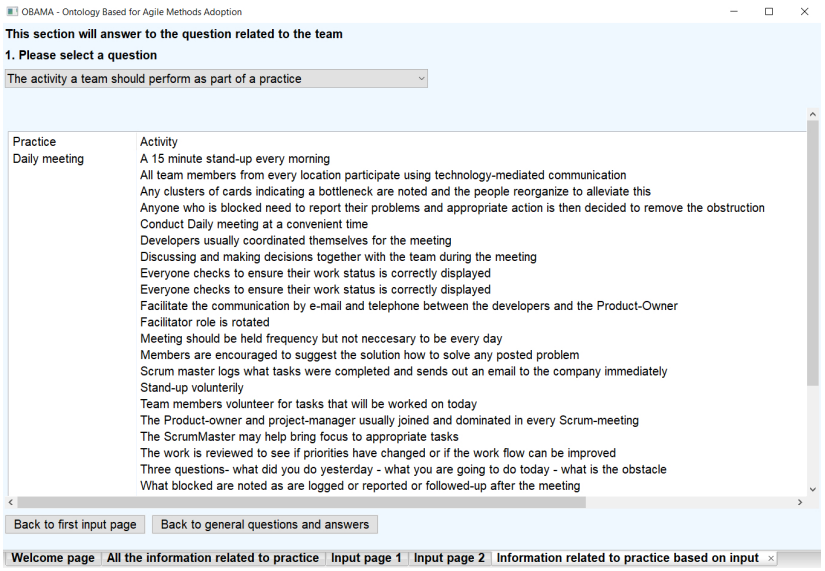


Fig. 8.15 Activities as part of *Daily meeting* listed by OBAMA-Tool

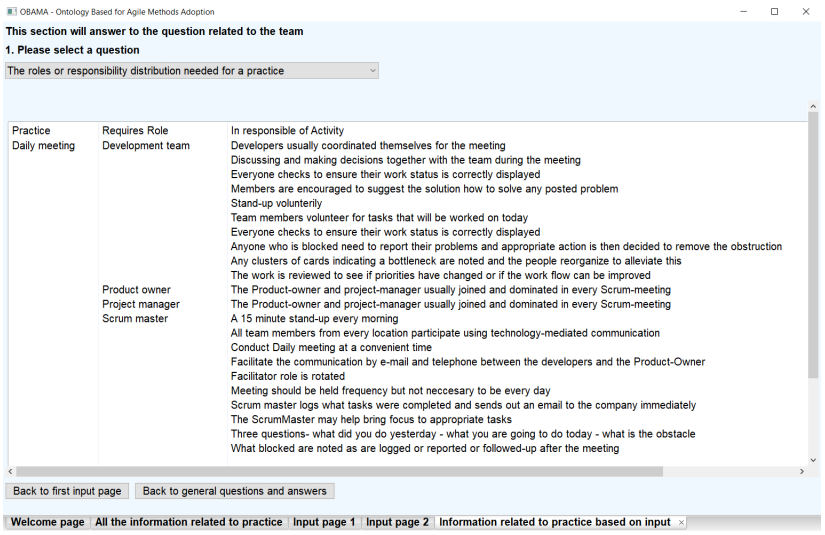


Fig. 8.16 Role required to adopting *Daily meeting* listed by OBAMA-Tool

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

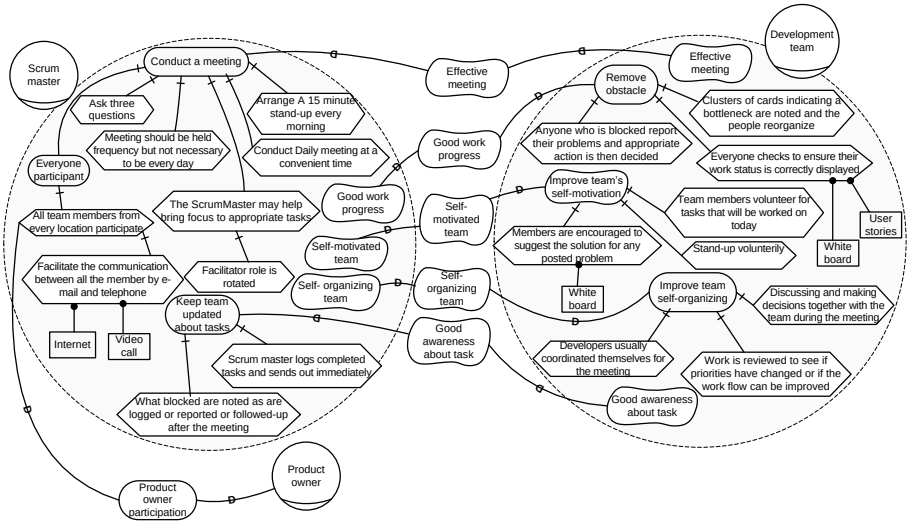


Fig. 8.17 Dependencies between roles to perform activities of *Daily meeting* visualized in iStar 2.0.

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.

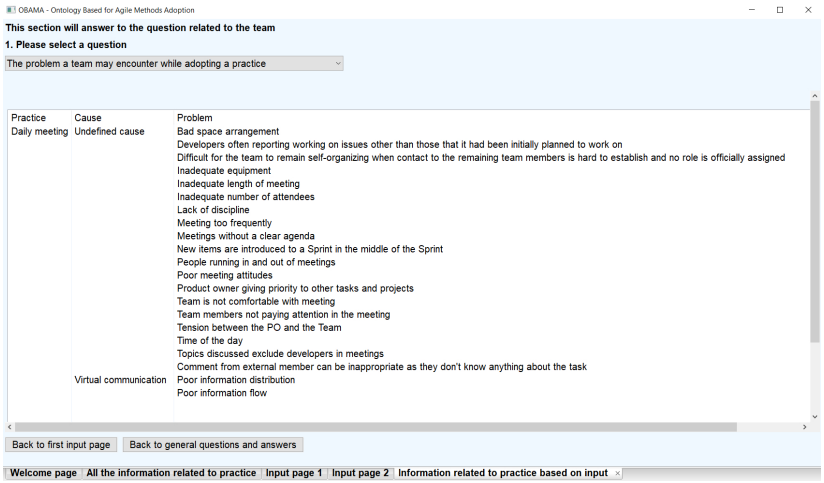


Fig. 8.18 Cause, Problems in *Daily meeting* listed by OBAMA-Tool

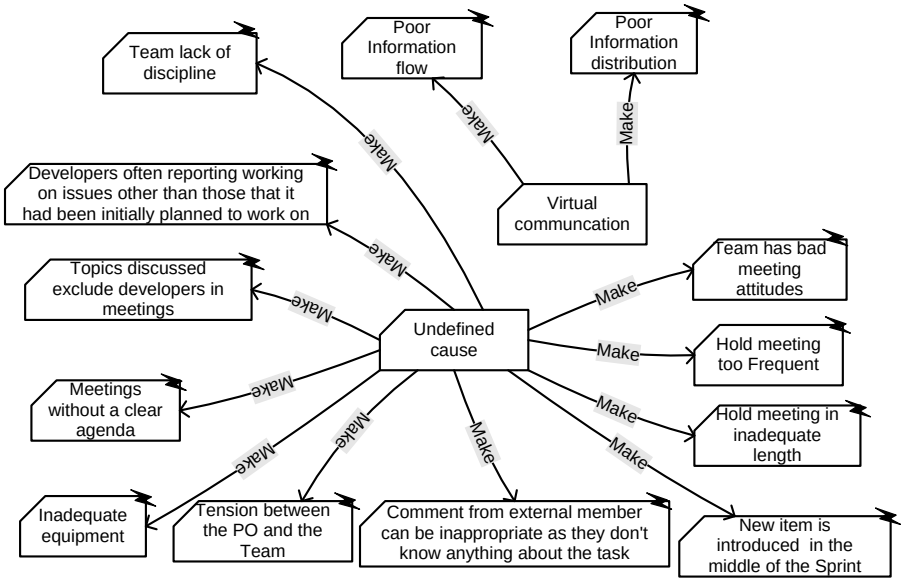


Fig. 8.19 Causes and Problems in *Daily meeting* visualized in iStar 2.0

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.

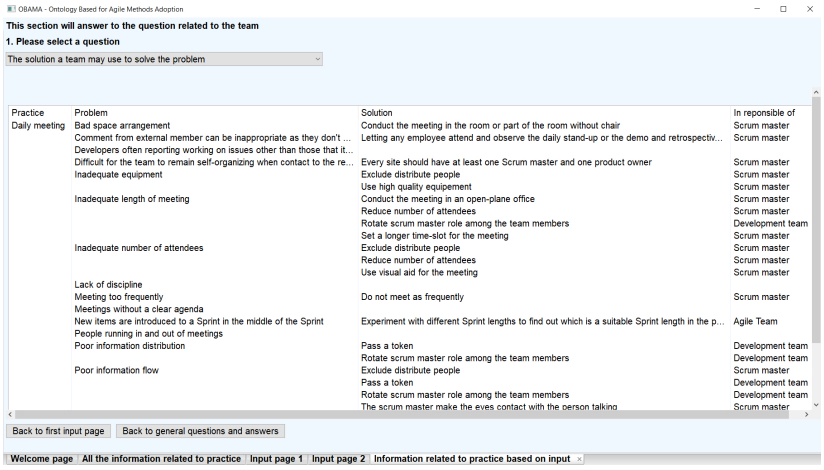


Fig. 8.20 Problems in *Daily meeting*, Solution and Role listed by OBAMA-Tool

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

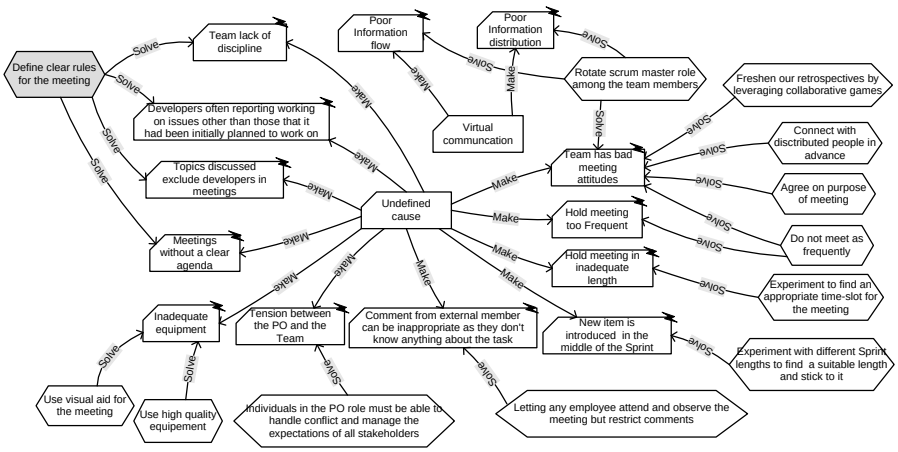


Fig. 8.21 Cause, Problems in *Daily meeting* and Solutions visualized in iStar 2.0

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.