Agile Methods Knowledge Representation for Systematic Practices Adoption

Soreangsey Kiv[1](✉), Samedi Heng[2], Manuel Kolp[1], and Yves Wautelet[3]

[1] LouRIM-CEMIS, UCLouvain, Louvain-La-Neuve, Belgium
{soreangsey.kiv,manuel.kolp}@uclouvain.be
[2] HEC Liège, Université de Liège, Liège, Belgium
samedi.heng@uliege.be
[3] KU Leuven, Leuven, Belgium
yves.wautelet@kuleuven.be

Abstract. The popularity of agile methods is constantly increasing. Information and feedback on how these frameworks were adopted can easily be found in academia and industrial knowledge bases. Such a collective experience allowed the development of many approaches in the aim of simplifying the adoption process and maximizing the chances of success. These approaches provide practitioners with guidelines to help them find the practice that suits their team best. Nonetheless, these approaches are not systematic and practitioners need to go through a long process. For instance, they need to identify the important situational factors that can have a positive/negative effect on the agile practice adoption. Available experiences thus require lots of effort to be discovered. This research proposes an agile methods knowledge representation using an ontology so that the knowledge and experience on agile adoption reported in literature may be reusable and systematic. Based on this model, added knowledge and inference rules, practitioners will systematically be able to decide which practice to select and adopt, i.e, for a given goal, practitioners can retrieve which practices to achieve; from a situation, teams can tell what can be harmful and what can be useful for adopting a practice or what problems they may encounter; etc.

Keywords: Agile methods · Agile practices · Ontology · Knowledge representation · Real case study

1 Introduction

Agile methods have been increasingly adopted by the software development industry (and others) due to their flexible features that allow to better handle the changes in requirements, to improve team's productivity and align to the business needs. As no method can be a one-size-fits-all, software development teams adopt agile methods differently, i.e., depending on their specific problems, resources, goals or expectations [4]. Many empirical studies of agile methods adoption have been published every year. The result from the Systematic Literature Review (SLR) in [5] points out that, in the methodological aspects used on agile methods tailoring research, 66.1% of their selected papers were empirical research. A simple search, also, in SpringerLink for "Daily Meeting" to this day, allows finding 1186 articles with 173 in the software engineering sub discipline. Some research papers describe their proper experience in deploying agile in their own organization, while some others discuss it based on empirical evidences collected from multiple cases. Those papers aim to share knowledge such as problems encountered, lessons learned, solutions found, etc., so that others can learn how to choose the right practices and avoid failures.

These experiences are extremely important and useful, yet time-consuming to collect and classify. Let us imagine that a development team aims to achieve a particular goal. How would they know which practices would help them to? In addition to "goal" to achieve, several variables have to be considered such as situation, project, budget, etc. which can also constrain the selection of a practice. Since the development process is complex and requires lots of effort, many teams decide recklessly to adopt specific agile methods or practices which are popular without considering any context-specific factors resulting in numerous agile adoption failures in the end [10].

To make the knowledge and experiences of the previous empirical studies easily accessible, [8] introduces a structured repository of Agile Method Fragments (AMF). This knowledge repository has been gathered through a systematic review of empirical studies on agile methods. For each AMF, the repository entry states the objectives the AMF aims to contribute to, and a set of requisites needed for its success. On top of that repository, the same authors also proposed a framework for evaluating the suitability of candidate method fragments prior to their adoption in software projects [9]. By linking (with contribution links such as help/harm) the situational factors to the requisite, practitioners can find out whether or not they have the chance to succeed with that practice adoption. Even though the repository and the framework can help practitioners to save much effort in understanding agile practices and their suitability, it is yet inefficient and not systematic enough. In order to use this framework, practitioners are expected to know what the situational factors affect the adoption. In addition, they have to figure out by themselves what parts are considered as helpful or harmful to the requisites and practices.

We argue that a better and efficient solution would be a system which can list out goals achieved by a practice, problems that may be encountered from a given situation and what the team needs to do to solve/avoid problems etc. The answers given by the system to these questions must be generated from the previous experiences of agile practitioners. This paper proposes using an "Ontology" to represent and store all these knowledge items of agile methods or practices adoption, reported in literature. Our goal is to make the existing experience reusable in a systematic manner.

This paper is organized as follows. Section 2 presents the research protocol we applied to achieve our research objective. Section 3 provides the detail of our ontology creation as well as the final ontology model in the form of a UML class diagram. Next, Sect. 4 provides the inference rules we have created for our ontology. The procedure of collecting case studies is described in Sect. 5. Section 6 provides an illustrative example of how to use our ontology when adopting an agile practice in a systematic manner. Finally, we conclude, discuss the limitations of and elaborate on future research directions of the paper in Sect. 7.

2 Research Methodology

Figure 1 depicts the research protocol we applied. We started by building the ontology which basically follows the methodology proposed in [17]. It consists of seven steps: *(1) Determining the domain and scope of the ontology*, *(2) Considering reusing existing ontologies*, *(3) Enumerating important terms in the ontology*, *(4) Defining the classes and the class hierarchy*, *(5) Defining the properties of classes slots*, *(6) Defining the facets of the slots*, and *(7) Creating instances.* The description of each step can be found in [17]. Due to limited space here, we merged step 4, 5 and 6 in Fig. 1. We, however, followed those three steps to create our ontology.

Since we need data from real case studies to build an evidence-based ontology for agile methods adoption, the process for *collecting real case studies* is also included into our research protocol. These case studies allow us to enumerate extra concepts and relationships and it also serves as data input for knowledge creation. The process of building our ontology is iterative and incremental [17]. It means that each case study from data collection was fed into the model for revising and refining the model. We repeated steps 3 to 7 until obtaining a consistent model which fits well with a representative amount of selected case studies (see Sect. 3). Two additional steps follow: *Building Inference Rules* (see Sect. 4) and *Validation Scenario* (see Sect. 6). The former aims at systematically discovering more relationships and the latter aims at providing a feasibility study, as a validation case of our approach.

3 Building the Agile Methods Ontology Model

This section describes how our ontology was built. As mentioned earlier, we started with determining the domain and scope of our ontology. We then discuss about existing ontologies, followed by how we enumerated terms to build our model. Next, we present our ontology model in the form of a UML class diagram. Finally, we describe how to insert the knowledge into the model.

3.1 Determining the Domain and Scope of the Ontology

The scope of the ontology presented here is limited to concepts, relationships and knowledge extracted from experience reported in research papers about adopting agile methods or practice for software development project. We aim at demonstrating the advantages of using the ontology for helping agile practitioners in

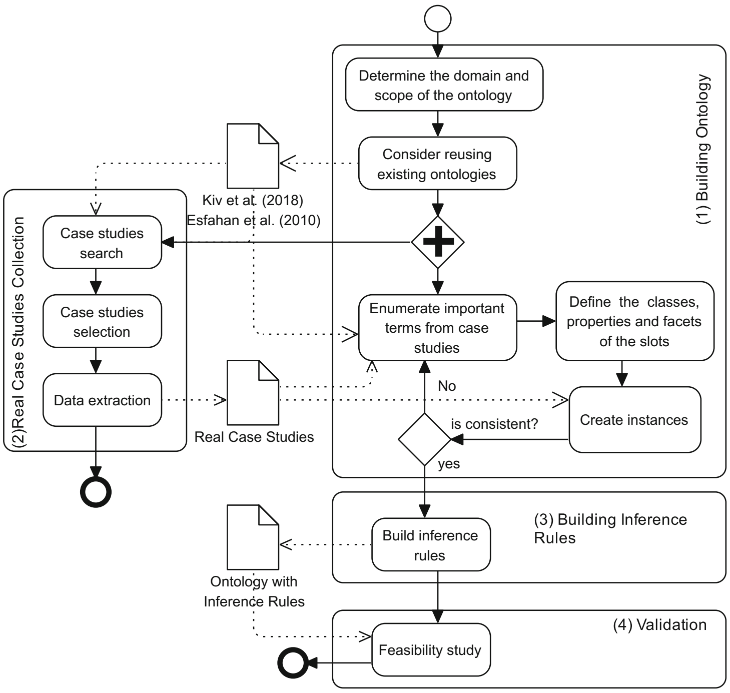

Determine the domain and scope of the ontology
Consider reusing existing ontologies
(1) Building Ontology
Kiv et al. (2018)
Esfahan et al. (2010)
(2)Real Case Studies Collection
Case studies search
Case studies selection
Data extraction
Real Case Studies
Enumerate important terms from case studies
Define the classes, properties and facets of the slots
Create instances
No
is consistent?
yes
(3) Building Inference Rules
Build inference rules
Ontology with Inference Rules
(4) Validation
Feasibility study

Fig. 1. Research protocol.

selecting and adopting agile practices in a systematic manner. Our preliminary study focuses on using the ontology to represent the knowledge and answer the following questions:

- *Q1: What objectives/goals can be achieved by an agile practice?*
- *Q2: What agile values and principles can be achieved by adopting a practice?*
- *Q3: What activities are part of a practice and need to be performed by the team?*
- *Q4: What are the requisites to successfully adopt a practice?*
- *Q5: What can be harmful when adopting a practice?*
- *Q6: What can be useful when adopting a practice?*
- *Q7: What kind of problems may a team encounter?*
- *Q8: What can be the solutions to a problem?*
- *Q9: What roles or responsibility distribution are needed for each practice?*
- *Q10: What are the artifacts required for a practice?*

Before building the ontology, we also considered reusing existing ones which can be found in specific libraries such as COLORE [6], DAML [2] and Protégé [3]. However, as mentioned, none of them is related to agile practices selection or adoption. We thus needed to build the model from scratch.

3.2 Enumeration of Important Terms

Three main resources help us to enumerate important terms in the ontology: (1) the repository proposed in [8], (2) the influence of the agile manifesto over agile practice selection studied in [13], and (3) the real case studies collected in research community (see Sect. 5).

We must admit that the repository proposed by [8] has inspired us in creating this ontology model. In their repository, each agile method fragment is linked to the objectives/goals it aims to contribute to and a set of requisites needed for its success. Then, the suitability of each fragment is linked to the situation of the team. For example, based on their repository, the goal of conducting "Daily Meeting" is to improve "Quality of Communication" and to conduct "Daily Meeting" successfully, it requires an "effective meeting". An "effective meeting" is suitable for the team that has a "highly available Scrum Master". Even though [8] does not give any clear definition of the concept "agile method fragment", based on our understanding from data in their repository, the authors refer this concept to "agile practice". Therefore, we use the term "practice" in our research. From these, we gathered some terms, which will then become classes, including: **practice**, **goal**, **requisite** and **situation**.

In [13] shows the importance of agile manifesto, i.e., agile values and principles, in adopting agile methods, and in [12] explains its relationship with practices. Understanding the agile manifesto allows us to know why we want to adopt an agile practice. In other words, adopting a practice can achieve the goals of adopting agile methods defined in the agile manifesto. For instance, to achieve the principle "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation", "Daily Meeting" is a suitable practice. In addition, by knowing what agile value and principle a team can achieve, they can measure their level of agility. Thus, we added **value**, and **principle** to the model.

Practice, **goal**, **requisite**, **situation**, **value**, and **principle** are the starting terms of our ontology creation. Then, to be able to answer to the questions in Sect. 3.1, we refined our model based on the result from the case studies. **Activity**, **problem**, **solution**, **role** and **artifact** are thus the extra classes we added to store the extracted information. To differentiate each case study, we added another class **Team** and linked to **goal**, **practice**, **situation**, **problem**, **activity** and **solution**.

3.3 Class and Relationship

After enumerating all the terms and incremental refinement, we built the ontology model with Protégé[1]. The final model is illustrated as a UML class diagram in Fig. 2. We describe the main concepts and relationships as follows:

– **Value:** refers to the agile values as defined in the agile manifesto. Based on [12], agile value is *contributed by* the principle;

[1] https://protege.stanford.edu/.

- **Principle:** refers to the agile principle as defined in the agile manifesto and it *contributes to* agile value;
- **Goal:** is the objective that *belongs to* a team in adopting agile methods. A goal can be *achieved by* conducting agile practices and achieving this goal can *contribute to* the agile principle or another goal;
- **Practice:** refers to an agile practice. It is *adopted by* the team, is *composed of* activities and allows the team to *achieve* the goal. Conducting a practice may *require* a requisite and it can also *encounter* a problem;
- **Team:** refers to a software project team that *has* a specific situation and goal. They *adopt* agile practice and *perform* activities as part of a practice. While conducting an agile practice, a team may *encounter* a problem and, as a result, may *propose* a solution;
- **Situation:** is the state that *belongs to* a team which can affect practice adoption as it can *help* or *harm* the requisite of a practice. In our case, only the situations listed in [5] are taken into account. They are *Project type*, *business goals*, *complexity*, *team size*, *technology knowledge*, *user availability*, *requirements stability*, *organization size*, *culture*, *team distribution*, *management support*, *degree of innovation*, *previous projects*, *maturity level*, *domain knowledge*, *project budget*, *communication* and *type of contract*;
- **Activity:** is *performed by* a team as *part of* the practice. For instance, "15 min meeting every morning" is a part of the "Daily Meeting". Performing an activity can *cause* a problem, *help* or *harm* a requisite and it may also *require* a role or artifact;
- **Requisite:** is the condition which is particularly *required by* a practice in order to successfully adopt it. For instance, conducting a "Daily Meeting" requires "ease of communication" and "everyone's participation". The requisite can be *helped* or *harmed by* team situation or activity. It can also *require* a role, artifact or other requisites;
- **Problem:** is the problem *faced by* team and practice while adopting a practice. For instance, one of the problems faced by a team described in [20] when adopting "Daily Meeting" was "starting promptness as the meetings did not start on time". Problem can be *caused by* a situation, activity or other problem. Some problems can be *solved by* the solution;
- **Solution:** is the solution *proposed by* team in order to *solve* the problem. It may *require* a role or artifact;
- **Role:** is the role *required by or responsible for* an activity, solution or requisite;
- **Artifact:** it is the artifact *required by* an activity, solution or requisite.

The relationships described above are only those made between classes which were manually built. In the ontology, we can discover more of them from reasoning using inference rules. They are listed in Table 2 (Sect. 4).

Class Hierarchy: One of the decisions to make during modeling is when to introduce a new class or when to represent the distinction through different property values [17]. For instance, there are seventeen different types of **Situation**; in line with [17], since each type has a different effect to the **Requisite**,

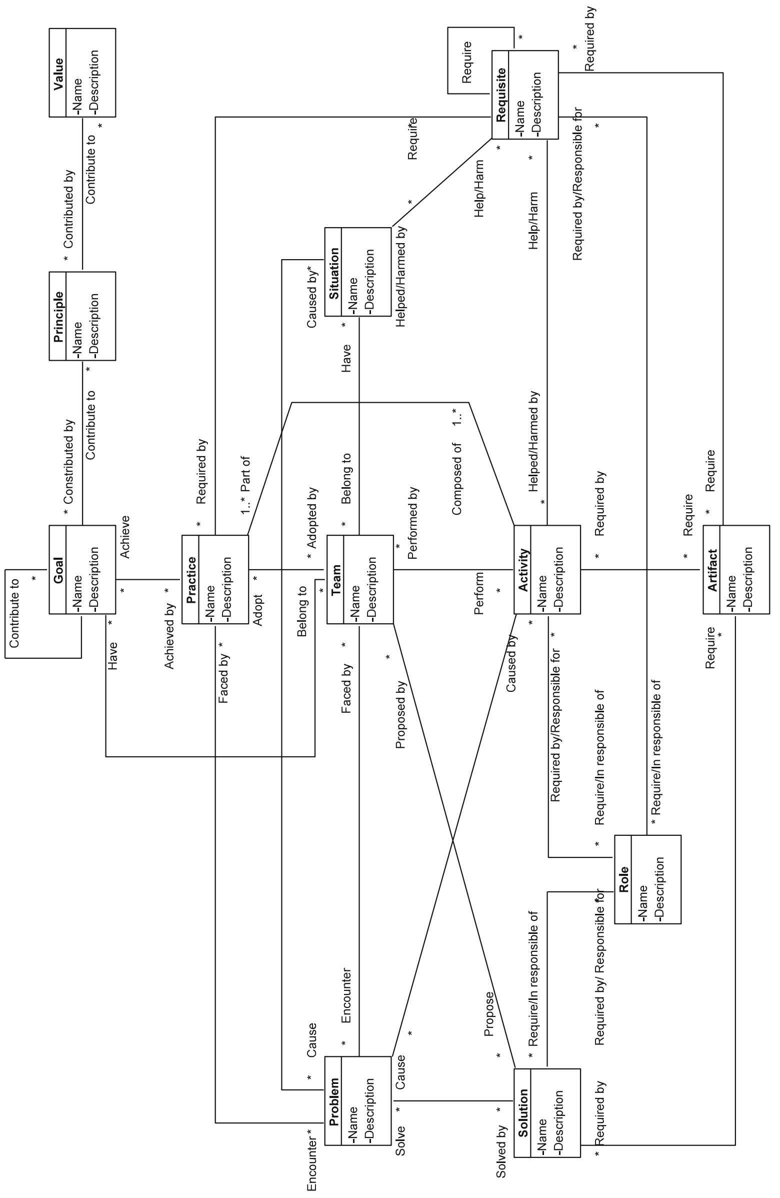


Fig. 2. An evidence-based ontology for agile methods adoption.

we thus create a subclass for each of them in our ontology model. To simplify the representation, we excluded these 17 sub-classes from Fig. 2.

Property: There are two types of property: data property and object property. Data property links individuals–i.e., instances and data values. Object property links individuals and individuals. Both links are built in the form of "Domain - data/object property - Range". For instance, the link "Practice - Name - String" means that, data property Name has Practice as domain and String as range. Another example, "Practice - Achieve - Goal" has Practice as the domain and Goal as the range of object property Achieve.

Every class in our model has only two data properties–i.e., Name and Description. They are the only common things to describe each class by agile practitioners. Their type is String. The domain and range of each object Property were built based on their relationships as in Fig. 2.

3.4 Instances Creation

In this section, we explain how data extracted from real case studies were inserted into our ontology. For illustration, we take a partial data extracted from a selected paper ([20] in the references at the end of the paper). The paper is about a case study of a software development project having three distributed teams – two are located in Norway and the other one is located in Asia. All teams have used Scrum with all the recommended practices for more than two years. Based on their experience, having distributed team causes some problems while conducting "Daily Meeting", such as starting promptness and information distribution.

To insert knowledge into the ontology we need to (1) analyze the description to know what should be created as individuals and in which class, (2) create individuals, and after that (3) connect the individuals by adding the data and object property to each individual. Table 1 shows the individuals and links we created for this case.

Table 1. An instance creation based on a case study.

Class name: individual	Object property	Class name: individual
Team:team1	Have	Situation:Distributed_team
Team:team1	Have	Situation:2_years_agile_experience
Team:team1	Adopt	Practice:Daily_meeting
Team:team1	Encounter	Problem:Starting_promptness
Team:team1	Encounter	Problem:Information_distribution
Practice:Daily_meeting	Encounter	Problem:Starting_promptness
Practice:Daily_meeting	Encounter	Problem:Information_distribution
Situation:Distributed_team	Cause	Problem:Starting_promptness
Situation:Distributed_team	Cause	Problem:Information_distribution

4 Building Inference Rules

Inference is one of the techniques to improve the quality of data integration by discovering new relationships, automatically analyzing the content of the data, or managing knowledge [1]. A simple example of the inference can be: If a taxi driver must be an adult; so if someone is a taxi driver then she/he must be an adult. Another simple example related to agile practice can be: if a team adopts a practice and that practice achieves a goal, we can infer that the team achieves that goal.

It is possible to build any relationship directly in the ontology but this will only weight down and complicate the model. Also, without the inference rule, we cannot discover any relationship between instances more than what we manually insert. That is not the efficient way of using knowledge. Therefore, if a relationship can be discovered by reasoning, we use the inference rule. Table 2 lists all the inference rules we have built in our ontology to discover more relationships in order to answer the questions in Sect. 3.1.

Similarly to ontology creation, there are different ways in writing inference rule. For instance, "If a problem is caused by an activity and that activity is part of a practice $\rightarrow$ that practice encounters that problem" can be written as "If a practice is composed of an activity and that activity causes a problem $\rightarrow$ that practice encounters that problem". However no repetitive inference rule should allow answering the same question.

5 Case Studies Data Collection

Following the procedure of ontology creation, we need to repetitively create the model and feed the data to see whether or not it can represent the knowledge we want to use in the future. Actually, there is no way to validate the model because new case studies keep coming in and the model can always be improved over time. The best we can do in this paper is to feed a good amount of case studies and try to answer our predefined questions.

We decided to take ten different case studies. For diversity, we took two cases for each of the five most commonly used agile practices based on the 12th VersionOne agile survey. They are *Daily stand-up*, *Sprint/iteration planning*, *Retrospectives*, *Sprint/iteration review*, and *Short iterations and release planning.*

To collect the documents that report about applying a specific agile practice in real projects, we basically followed the steps for conducting SLR described in [13]. We briefly describe those steps hereafter:

- **Keyword:** Even though we only took two cases for each practice, we tried to retrieve all the papers related to each practice adoption to check and select the best two. Keywords are thus the name of each practice, which are "daily standup", "sprint planning OR iteration planning", "retrospectives", "sprint review OR iteration review", "short iterations" and "release planning".

Table 2. Inference rules for answering questions in Sect. 3.1.

Question	Inference rules
Q1	***R1***: *If a practice is composed of an activity and that activity achieves a goal* $\rightarrow$ *that practice achieves that goal.*
Q2	***R2***: *If a practice achieves a goal and that goal contributes to a principle* $\rightarrow$ *that practice achieves that principle.*
	R3: *If a practice achieves a principle and that principle contributes to a value* $\rightarrow$ *that practice achieves that value.*
Q2	*Can be discovered with direct relationship.*
Q4	***R4***: *If a practice requires a requisite and that requisite is helped by another requisite/situation/activity* $\rightarrow$ *that practice requires all of that requisite/situation/activity.*
Q5	***R5***: *If a situation/activity harms a requisite and that requisite is required by a practice* $\rightarrow$ *that situation/activity harms that practice.*
	R6: *If a team has a situation and that situation harms the requisite* $\rightarrow$ *that team harms that requisite.*
	R7: *If a team performs an activity and that activity harms the requisite* $\rightarrow$ *that team harms that requisite.*
Q6	***R8***: *If a situation/activity helps a requisite and that requisite is required by a practice* $\rightarrow$ *that situation/activity helps that practice.*
	R9: *If a team has a situation and that situation helps the requisite* $\rightarrow$ *that team helps that requisite.*
	R10: *If a team performs an activity and that activity helps a requisite* $\rightarrow$ *that team helps that requisite.*
Q7	***R11***: *If a practice is composed of an activity and that activity causes a problem* $\rightarrow$ *that practice encounters that problem.*
	R12: *If a team performs an activity and that activity causes a problem* $\rightarrow$ *that team encounters that problem.*
	R13: *If a team has a situation and that situation causes a problem* $\rightarrow$ *that team encounters that problem.*
	R14: *If a team encounters a problem_1 that causes another problem_2* $\rightarrow$ *that team encounters the problem_2.*
Q8	*Can be discovered with direct relationship.*
Q9	***R15***: *If a person is responsible for an activity* $\rightarrow$ *that person is required for that activity.*
Q9& Q10	***R16***: *If a practice is composed of an activity and that activity requires a role* $\rightarrow$ *that practice requires that role.*
	R17: *If a team performs an activity and that activity requires a role/artifact* $\rightarrow$ *that team requires that role/artifact.*

– **Search Engines:** We took the formal data from well-known digital libraries in the field of software engineering: IEEEXplore, ScienceDirect, ACM Digital library and SpringerLink. We set the publication years to between 2000 and

2018, the field to Software Engineering, and the search terms matching title of the paper, keywords or abstract.

- **Selection Criteria:** With a big list of papers related to each practice, we did an abstract screening then a full-text screening with the following criteria:
  - Empirical study or research study with case study validation related to agile methods or agile practices usage or adoption;
  - Paper that has a significant discussion related to the keyword practice. As the result, it must describe the usage experience and/or the lesson learned and/or the problem and/or the challenge and/or the solution to the problem;
  - Paper with a good description of team situations and goal.
- **Data Extraction:** We extracted data based on the questions defined in Sect. 3.1. Basically, we tried as much as we could to extract the following information from each paper: **goal**, **activity**, **requisite**, **situation**, **problem**, **solution**, **role** and **artifact**.

While many of them meet the criteria, we decided to choose the two most descriptive cases, the ones which can answer best the questions in Sect. 3.1. They are Stray, et al. [20] and Moe and Aurum [15] for **Daily meeting**, Berteig, M. (2008) and Ochodek, M. & Kopczyńska (2018) for **Short iteration**, Gregorio [11] and Moe, et al. [16] for **Sprint planning**, Maham [14] and Paasivaara and Lassenius [18] for **Sprint retrospective**, and Santos [19] and Eloranta [7] for **Sprint review**.

6 Feasibility Study

Once the ontology model was built, and knowledge and inference rules added, the model is ready to be used. In this section, we provide an illustrative example of how to use our ontology when adopting an agile practice in a systematic manner.

As an illustrative scenario, consider an agile software development team which is assigned to develop a mobile application. The team has the following situation: (1) Some of team members are new and others have an extensive experience with mobile application development. (2) The team is working in two locations and only one team has direct access to their clients. (3) All of them are neophytes to distributed development. (4) Some of them are new to agile methods and others have been developing some projects with Scrum for a few years. The team decides to use Scrum. The Scrum Master understands that bad communication can cause some problems in adopting "Daily Meeting". Therefore, his goal is to make communication effective. He is wondering if there are reports or documents discussing about the problems related to communication encountered by a distributed team when adopting "Daily Meeting". What are their solutions for addressing these problems? Such information is very useful for the Scrum Master and may inspire him to adopt "Daily Meeting" successfully.

With the same Protégé Tool only requires four simple steps in order to get the answers. (1) Creating a new individual to represent development team, (2)

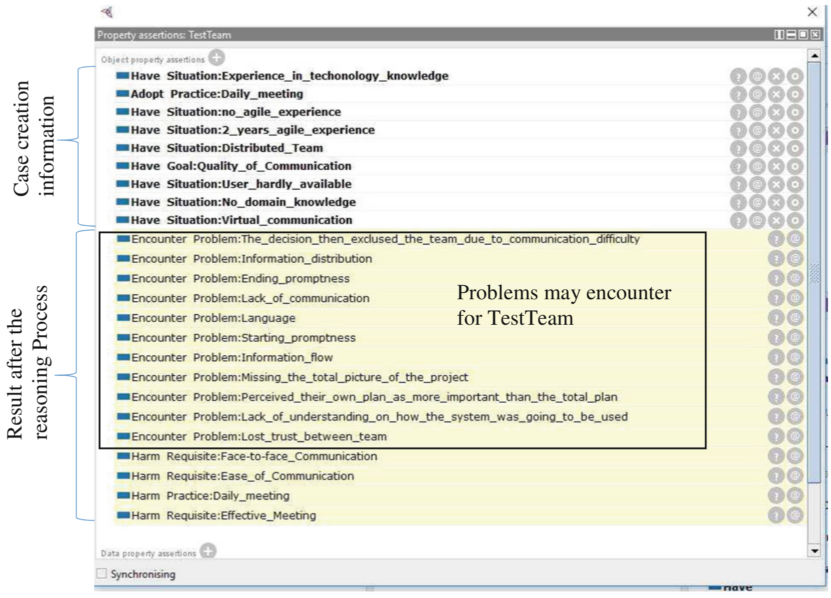
Table 3. Relationship in ontology format for feasibility scenario.

Class name: individual	Object property	Class name: individual
Team:TestTeam	Have	Situation:Distributed_team
Team:TestTeam	Have	Situation:2_years_agile_experience
Team:TestTeam	Have	Situation:No_agile_experience
Team:TestTeam	Have	Situation:User_hardly_available
Team:TestTeam	Have	Situation:No_domain_knowledge
Team:TestTeam	Have	Situation:Experience_in_technology_knowledge
Team:TestTeam	Have	Situation:Virtual_communication
Team:TestTeam	Have	Goal:Quality_of_Communication
Team:TestTeam	Adopt	Practice:Daily_meeting

connecting their team individual with the existing individuals which match the team's situation and goal, (3) executing the reasoning to get all the individuals linked to the team, (4) using query to get more answers to the question described in Sect. 3.1.

With the above scenario, we created individual "Team:TestTeam" to represent the development team. Then, we linked TestTeam to different individuals based on the team's situation and goal as in Table 3.

Next, we started the reasoner to discover more individuals linked to the TestTeam. At once, all the inference rules in Table 2 were executed. Among


Case creation information
Result after the reasoning Process
Problems may encounter for TestTeam

Property assertions: TestTeam

Object property assertions

```
Have  Situation:Experience_in_techonology_knowledge
Adopt  Practice:Daily_meeting
Have  Situation:no_agile_experience
Have  Situation:2_years_agile_experience
Have  Situation:Distributed_Team
Have  Goal:Quality_of_Communication
Have  Situation:User_hardly_available
Have  Situation:No_domain_knowledge
Have  Situation:Virtual_communication
Encounter  Problem:The_decision_then_exclused_the_team_due_to_communication_difficulty
Encounter  Problem:Information_distribution
Encounter  Problem:Ending_promptness
Encounter  Problem:Lack_of_communication
Encounter  Problem:Language
Encounter  Problem:Starting_promptness
Encounter  Problem:Information_flow
Encounter  Problem:Missing_the_total_picture_of_the_project
Encounter  Problem:Perceived_their_own_plan_as_more_important_than_the_total_plan
Encounter  Problem:Lack_of_understanding_on_how_the_system_was_going_to_be_used
Encounter  Problem:Lost_trust_between_team
Harm  Requisite:Face-to-face_Communication
Harm  Requisite:Ease_of_Communication
Harm  Practice:Daily_meeting
Harm  Requisite:Effective_Meeting
```

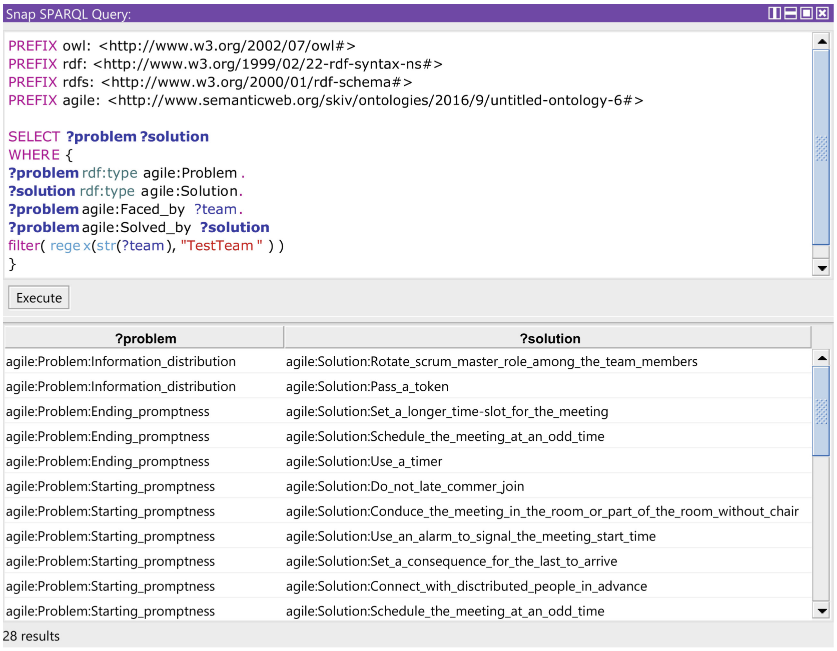
Data property assertions

Synchronising

Fig. 3. Case result: problems encountered by team.

these 17 rules, R6, R7, R9, R10, R12, R13, R17 are related to the Team. That is why, from the individual TestTeam, the Scrum Master can have the answers related to problems that his team may encounter and to the situation of the team that helps and harms the requisite of the "Daily Meeting". Figure 3 exposes the result from the reasoning. As expected, the TestTeam may encounter multiple problems since its situations are harmful for the requisite as well as the "Daily Meeting".

Since **Solution** is not linked to the **Team**, in order to get answers, it requires to run a query. In our case, we used SPARSQL. The query and result shown in Fig. 4 are the solutions to the problem that the TestTeam may encounter. As an example, two solutions may address the problem "Information distribution". They are "Rotate scrum master role among the team members" and "Pass a token". More answers for the ten predefined questions in Sect. 3.1 can be found at https://goo.gl/sSBAZo.



```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX agile: <http://www.semanticweb.org/skiv/ontologies/2016/9/untitled-ontology-6#>

SELECT ?problem ?solution
WHERE {
?problem rdf:type agile:Problem .
?solution rdf:type agile:Solution.
?problem agile:Faced_by ?team.
?problem agile:Solved_by ?solution
filter( regex(str(?team), "TestTeam" ) )
}
```

?problem	?solution
agile:Problem:Information_distribution	agile:Solution:Rotate_scrum_master_role_among_the_team_members
agile:Problem:Information_distribution	agile:Solution:Pass_a_token
agile:Problem:Ending_promptness	agile:Solution:Set_a_longer_time-slot_for_the_meeting
agile:Problem:Ending_promptness	agile:Solution:Schedule_the_meeting_at_an_odd_time
agile:Problem:Ending_promptness	agile:Solution:Use_a_timer
agile:Problem:Starting_promptness	agile:Solution:Do_not_late_commer_join
agile:Problem:Starting_promptness	agile:Solution:Conduce_the_meeting_in_the_room_or_part_of_the_room_without_chair
agile:Problem:Starting_promptness	agile:Solution:Use_an_alarm_to_signal_the_meeting_start_time
agile:Problem:Starting_promptness	agile:Solution:Set_a_consequence_for_the_last_to_arrive
agile:Problem:Starting_promptness	agile:Solution:Connect_with_disctributed_people_in_advance
agile:Problem:Starting_promptness	agile:Solution:Schedule_the_meeting_at_an_odd_time

28 results

Fig. 4. Case result: proposed solution.

7 Conclusion and Future Work

In this paper, we presented the creation and uses of an ontology to support knowledge representation aiming at recycling agile adoption experience. It has been built on the basis of knowledge extracted from empirical evidence reported in existing literature. Seventeen inference rules have been added to systematically discover more relationships among concepts in our ontology.

Through knowledge representation, practitioners using it dispose of a tool to systematically and effectively support their own agile adoption. By using Protégé, in just four simple steps, they can systematically answer common questions related to the selection and adoption of a particular agile practice. Examples include determining what goal can be achieved by adopting a practice; what can be harmful and what can be useful for adopting a practice into a particular situation; what problem may be encountered and what does the team need to do to solve/avoid that problem, etc. To get answers, agile practitioners simply need to select the existing situations in the model that match their own. In addition, as the answers are generated from previous experiences, they would be very helpful and pragmatic.

The main limitation at this stage of this research concerns the handling of conflict situations. For instance, the feasibility scenario allows team members to be neophyte or expert in agile methods. In this case, our model cannot make a conclusion for such a mixed situation. It can only tell what is helpful and what is harmful about each situation independently. Another limitation concerns the fact that some of the answers cannot be generated by the reasoning – i.e, the model cannot provide what situations are considered as harmful if agile practitioners choose to adopt a specific practice. It will list all the problems caused by the situation regardless of the practice adopted by the agile practitioners. Getting such answer requires using a query too complex for agile practitioners in learning. Finally, the included knowledge is still limited; with only ten case studies, there are situations which the model cannot answer.

For addressing the limitation, we plan to add more knowledge into our ontology in the near future. Within the SLR process, we selected in total more than 100 case studies for the five most commonly used practices. We hope to improve our model by adding not only these knowledge but also additional inference rules. Moreover, we also plan to build a user friendly Computer-Aided Software Engineering (CASE) Tool available for agile practitioners for using and encoding knowledge themselves so that the knowledge base would be increased. Finally, a real experimentation with agile software development teams will be conducted to get their feedback on the usefulness of our approach.

References

1. Inference. https://www.w3.org/standards/semanticweb/inference
2. DAML ontology library (2004). http://www.daml.org/ontologies
3. Protege ontology library (2018). https://protegewiki.stanford.edu/wiki/Protege_Ontology_Library
4. Abbas, N., Gravell, A.M., Wills, G.B.: Using factor analysis to generate clusters of agile practices (a guide for agile process improvement). In: 2010 AGILE Conference, pp. 11–20. IEEE (2010)
5. Campanelli, A.S., Parreiras, F.S.: Agile methods tailoring-a systematic literature review. J. Syst. Softw. **110**, 85–100 (2015)

6. COLORE: Semantic technologies library. http://stl.mie.utoronto.ca/colore
7. Eloranta, V.P., Koskimies, K., Mikkonen, T.: Exploring ScrumBut—an empirical study of scrum anti-patterns. Inf. Softw. Technol. **74**, 194–203 (2016)
8. Esfahani, H.C., Yu, E.: A repository of agile method fragments. In: Münch, J., Yang, Y., Schäfer, W. (eds.) ICSP 2010. LNCS, vol. 6195, pp. 163–174. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14347-2_15
9. Esfahani, H.C., Yu, E., Cabot, J.: Situational evaluation of method fragments: an evidence-based goal-oriented approach. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 424–438. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13094-6_33
10. Fitzgerald, B., Russo, N., O'Kane, T.: An empirical study of system development method tailoring in practice. In: ECIS 2000 Proceedings, p. 4 (2000)
11. Gregorio, D.D.: How the business analyst supports and encourages collaboration on agile projects. In: 2012 IEEE International Systems Conference (SysCon), pp. 1–4. IEEE (2012)
12. Kiv, S., Heng, S., Kolp, M., Wautelet, Y.: An intentional perspective on partial agile adoption. In: Proceedings of the 12th International Conference on Software Technologies - Volume 1, ICSOFT, pp. 116–127. INSTICC, SciTePress (2017)
13. Kiv, S., Heng, S., Kolp, M., Wautelet, Y.: Agile manifesto and practices selection for tailoring software development: a systematic literature review. In: Kuhrmann, M., et al. (eds.) PROFES 2018. LNCS, vol. 11271, pp. 12–30. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03673-7_2
14. Maham, M.: Planning and facilitating release retrospectives. In: Agile 2008 Conference, pp. 176–180. IEEE (2008)
15. Moe, N.B., Aurum, A.: Understanding decision-making in agile software development: a case-study. In: 34th Euromicro Conference on 2008 Software Engineering and Advanced Applications, SEAA 2008, pp. 216–223. IEEE (2008)
16. Moe, N.B., Aurum, A., Dybå, T.: Challenges of shared decision-making: a multiple case study of agile software development. Inf. Softw. Technol. **54**(8), 853–865 (2012)
17. Noy, N.F., McGuinness, D.L.: Ontology development 101: a guide to creating your first ontology (2001)
18. Paasivaara, M., Lassenius, C.: Scaling scrum in a large globally distributed organization: a case study. In: 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE), pp. 74–83. IEEE (2016)
19. Santos, R., Flentge, F., Begin, M.-E., Navarro, V.: Agile technical management of industrial contracts: scrum development of ground segment software at the European Space Agency. In: Sillitti, A., Hazzan, O., Bache, E., Albaladejo, X. (eds.) XP 2011. LNBIP, vol. 77, pp. 290–305. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20677-1_21
20. Stray, V.G., Lindsjorn, Y., Sjoberg, D.I.: Obstacles to efficient daily meetings in agile development projects: a case study. In: 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 95–102. IEEE (2013)