

# Adopting Agile Methods: Can Goal-Oriented Social Modeling Help?

Hesam Chiniforooshan Esfahani  
Department of Computer Science  
University of Toronto  
Hesam@cs.toronto.edu

Jordi Cabot  
NRIA  
Ecole des Mines de Nantes  
jordi.cabot@inria.fr

Eric Yu  
Faculty of Information  
University of Toronto  
yu@ischool.utoronto.ca

## Abstract

*The heavy reliance on the human factor in agile methods poses new challenges for organizations intent on adopting them. Improper role assignment, neglected team dependencies, and overlooked required skills have all been reported as reasons for failures during the introduction of an agile method. Current process modelling languages are not designed for describing or analyzing such human-related issues, and thus, provide little assistance to organizations in the process of adopting an agile method. This paper advocates the use of goal-oriented modeling techniques for depicting social aspects of agile methods. These social models can be used to identify the key factors that contribute to the success or failure of an agile method, thus providing guidance early during the introduction of the method in an organization. The approach is illustrated using the Scrum process.*

## 1. Introduction

The introduction of a new software process in an organization is a difficult and often risky proposition. This is especially true when adopting agile processes since they usually imply radical departures from traditional team structures, cooperation styles and relationships [1], [2]. Experiences on the introduction of agile methods in different types of organizations highlighted the difficulty of instilling a clear and shared understanding of the new process among all team members [4]–[6].

In software process modeling, many types of process models have been introduced, focusing on different aspects of software processes (e.g. activity, product, resource, etc.) [7], [28], [29]. However, none of these commonly-used process descriptions focuses on the social aspects of a software process such as motivations for performing an activity, needed skills for

playing a role, or dependencies among team members. Therefore, current process modeling techniques provide inadequate assistance to organizations, willing to adopt an agile process, where a clear understanding of all these factors is key to successfully integrate agile practices within the organization.

This paper proposes to complement existing process modeling approaches with a new perspective aimed at describing and analyzing the social and human aspects of a software process. This approach makes use of the *i\** modelling framework [19] for visualizing and analyzing relationships among social actors, particularly how they depend on each other to achieve individual and team goals.

We believe this explicit representation of the social requirements (in terms of people, skills, dependencies and tasks) of a process facilitates its adoption and, furthermore, enables software companies to assess the chances of successfully enacting it by checking (prior to process adoption) whether the social aspects of the process will be a good fit for current team members. In this sense, an additional benefit of our approach is that it helps in the customization phase of software processes. Most organizations will need adaptations [10], rather than following the exact practices of a prescribed process (as in *If one fits all, it doesn't fit any*). Typically, adapted processes are obtained by way of tailoring existing methods [11], and should take into account the people aspects of both the target organization and the candidate software process. Refining the social model of a process with respect to the current situation of the organization facilitates the introduction of the process to the organization, and the identification of vulnerabilities that are specific to the organization.

As a running example, we will illustrate our proposal by focusing on the social aspects of Scrum - an agile process with emphasis on the importance of teamwork in software development [9]. We used the social model of Scrum to build a customized process for a classroom

case study, analyzed it to identify its key vulnerabilities, and evaluated process goals to predict to what extent they will be satisfied. We compared our model-based evaluation to what was observed in reality. The favorable comparison that our social models can indeed be used to predict the main (organizational) problems when adopting a new development process. The social models presented in the paper can be constructed using OpenOME <sup>1</sup>, an open source, Eclipse-based modeling tool to support *i\** modeling.

The rest of this paper is organized as follows. Section 2 motivates the need for social models of software processes. Section 3 presents a goal-oriented approach for the specification of such models. Section 4 presents the results of applying our approach on a case study. Finally, Section 5 discusses related work and Section 6 presents conclusions and future work.

## 2. Why social modeling?

The shift of concerns from technical to social issues in agile methods suggests the need for systematic ways of modelling and analyzing human-related aspects of agile methods [16]. This representation and analysis should be done by means of graphical models, since a visual representation of a software process (even if still requiring accompanying textual descriptions) is better when trying to understand a software process [12].

Current graphical depictions of agile methods are mostly informal diagrams or activity-oriented models, which focus on the activities and artifacts of the process. For instance, Figure 1 shows an informal diagram, and Figure 2 an activity-oriented model of the Scrum process. When these kind of process descriptions are used to describe an agile method, they still convey important information but, clearly, important aspects, particularly those concerning human relationships, are not represented and are therefore missed in the analysis. As a consequence, with current process modeling languages, the often crucial social aspects of agile methods would generally remain hidden in the textual descriptions and argumentations [15] of the process, which hinders their comprehension and systematic analysis.

To better encompass the social considerations that can make or break the enactment of an agile process, we need modeling techniques that can represent human related issues as required by the agile method. Such models can help organizations evaluate whether a process is socially compatible with their philosophy and inherent structures. Thus, the risk of adopting a

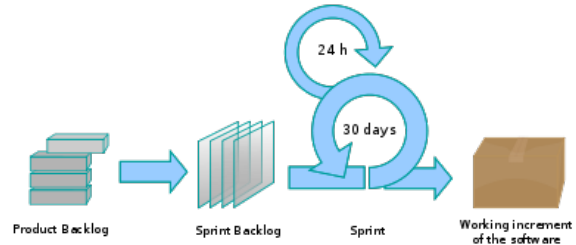


Figure 1. Informal SCRUM representation from [9]

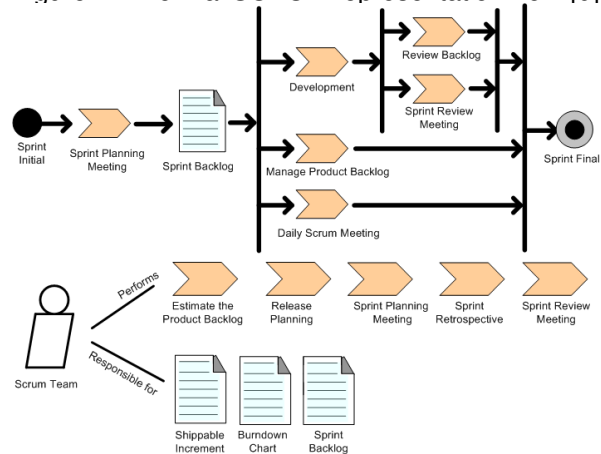


Figure 2. Activity-oriented representation of SCRUM (using SPEM 2.0 notation [13])

new process will be reduced, as the vulnerabilities of the process can be highlighted (and thus, minimized). For instance, when adopting an agile method, the organization and the team members participating in the new development process will have concerns such as:

- Which other actors will I depend on in order to succeed in my goals? What do I depend on them for?
- What are the process objectives that I have to fulfill?
- What are the skills required to perform each role?
- Which are the most critical roles? What are the consequences if these roles do not successfully fulfill their tasks?
- What are the rationales of performing certain development activities?

These questions (and many others that typically arise in a process-change scenario) cannot be easily answered by studying models such as those in Figures 1 and 2. Therefore, next sections introduce a social modeling perspective for software processes to complement the current description of software process and improve the chances of successfully adopting an agile process within an organization.

1. Available online at <https://se.cs.toronto.edu/trac/ome>

### 3. Goal-Oriented Social modeling of Software Processes

When adopting a social perspective, process roles become the central modeling construct. We treat roles as being intentional in that they have goals and strategic interests. Besides, process roles can have multiple dependencies to each other in order to collaboratively achieve their goals. These kinds of complex multi-role problems have been successfully addressed in the requirements engineering community by means of agent- and goal-oriented techniques (AGORE) such as  $i^*$  [19]. AGORE emphasizes the analysis of actors' goals and how they might be impacted by other actors' decisions. This analysis considers the collaboration of actors in reaching domain goals, through social chains of dependencies.

The  $i^*$  framework can be used to represent process roles (as a kind of actor in the  $i^*$  notation), their social dependencies, their goals, tasks, and resources. Goals can be hard-goals, representing the functional objectives, or softgoals, expressing qualitative objectives. One of the distinguishing capabilities of  $i^*$  modeling framework is its capability of modeling the different ways that a softgoal can be contributed (positively or negatively). This capability is quite helpful for modeling qualitative objectives of a social actor, since they might receive heterogenous contributions, showing the trade-offs that should be resolved prior to process adoption. For instance, for a *Project Manager* (a process actor), documentation (a process task) contributes positively to his/her softgoal *Facilitated Code Maintenance*, and at the same time negatively to the *Reduced Development Cost*. In [32] the  $i^*$  framework has been used to address the problem of trade-off analysis.

Therefore, we propose to make use of the  $i^*$  modeling framework to model the social perspective of a software process. This framework has been previously used for representing intentional aspects of organization-specific software processes [18]. The purpose of that work was to extract a descriptive process model of a software company to clarify the organizational complexities of actors' interactions. This work is intended to present prescriptive process models of a pre-defined software process, to facilitate its adoption and analysis by companies interested in it.

We will follow a two-step process. First, we will define a *Strategic Dependency* (SD) model. An SD model depicts the dependencies between the different (social) actors without delving into their internal intentions. Then, we will refine this model by defining a *Strategic Rationale* model to represent a more microprocess

view [3] of the agile method. SR diagrams model actors' internal goals, and their supporting rationales, that is, the combination of activities, artifacts, qualitative attributes, subgoals, and dependencies that help the actor to achieve its major goals. Once we have the models defined, it will be easy to highlight the major vulnerabilities of the process (important to pay special attention to them before adopting a process), and to answer different kinds of social/organizational questions about the process, as the ones pointed out in the previous section.

A complete description of the  $i^*$  notation is outside of the scope of the paper (see [19] for details) but Table 1 provides a summary of its main elements and our proposed interpretation in the context of social models of software processes. The graphical symbol corresponding to each element is shown in the legends accompanying the figures of this section.

Table 1. The  $i^*$  notations and their role in the modeling of social aspects of software processes

$i^*$ notation	Process Interpretation
Goal	Process functional objective (state that should be reached during process enactment) with clear-cut achievement criteria
Sofgoal	Process qualitative objective, including expected quality attributes of activities or products (e.g. <i>meetings be Effective</i> ), as well as needed skills and objectives with no clear-cut satisfiability
Task	Activities and practices prescribed by the process
Resource	Resources needed for, and artifacts developed during the enactment of process
Actor	Roles defined in a process, and agents who play those roles
Means-Ends Link	Alternating tasks for achieving a goal
Decomposition Link	Constituting elements of a task
Contribution Link	Effect of different elements on a softgoal
Dependency Link	Dependency relations between process actors

#### 3.1. Modeling intentional dependencies between actors

The SD model is used for this purpose. An SD model focuses on visualizing the roles defined by a process, and the dependencies among them. For instance, Scrum introduces three roles: Team, Scrum Master, and Product Owner. As an example, Figure 3 along with modeling these three roles, represents some dependencies that exist between them. For example, the Product Owner depends on Team to develop product increments, and Team depends on Scrum Master to remove its obstacles and to guide it in transition to the new method. (Although "Team" sounds to be a

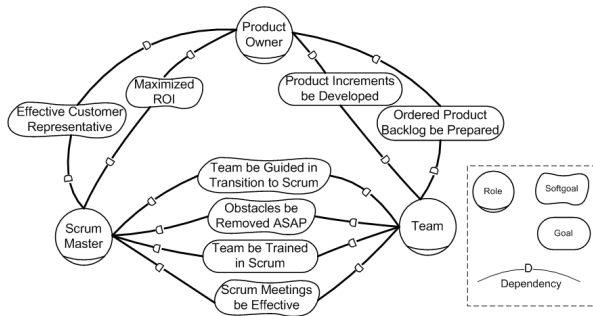


Figure 3. Actor dependencies in a Scrum process

position rather than a role, in order to follow the Scrum terminology, we avoided to change the role names, yet "Team Member" is what actually implied by "Team")

Apart from (soft)goal-dependencies, in which the dependee actor is in charge of attaining the (soft)goals delegated by the depender, an SD model can represent task- and resource-dependencies, respectively referring to tasks to be performed and resources to be provided by dependee on behalf of depender.

Systematic representation of social interdependencies can help organization members to have better understanding of the process, by clarifying their responsibilities and expectations. This alone can prevent some cases of process failure. For instance, it has been reported that one of the Scrum failure points happens when the Scrum Master does not remove the Team's obstacles [17]. In our model, it is clearly stated that this is a responsibility that the Scrum Master must take care of. Furthermore, modeling intentional dependencies also highlights the critical roles, which should be then carefully assigned to the most eligible members. Critical roles are those with notable number of incoming dependency links (e.g. Figure 3 accents the criticality of the Scrum Master role, as the number of its incoming dependency links is notably higher than the others). We would like to remark the importance of softgoals in social dependencies. Even if there is no clear-cut measurement scale to evaluate their success or failure, the dependee is supposed to do his/her best to satisfy the softgoal. This issue should also be carefully considered while assigning process roles to organizational actors.

Following the dependency relations, we may also detect how the poor performance of an actor, might affect the achievement of process goals and the functionality of other actors - what project managers seek to know [6]. Finally, dependencies assigned to a role also imply that the role should have certain skills to successfully fulfill the dependency relation. For instance,

the Scrum Master has to be an Scrum expert, since it is expected to teach Scrum to the team members. This issue will be further elaborated while modeling actors' goals and rationales.

### 3.2. Modeling Actors' Goals and Rationales

A more detailed analysis of a software process and the implications of poor performance of actors, required skills,... requires a more micro-process view of the process [3]. To do so, we refine the previous SD model with SR models that include all internal activities, artifacts, qualitative attributes, subgoals, and dependencies, which help process actor to achieve their major goals. All these elements are specified within the boundaries (dashed-line circles) of the actor symbol corresponding to the role.

As an example, Figure 4 depicts an SR model of Scrum, focusing on the Scrum Master. It is easy to see that this SR model contains much more detailed information about the Scrum process. It adds new internal goals that are required to satisfy the external dependencies for the actor (e.g. *Facilitated Customer/Team Collaboration*), explicit additional skills (e.g. softgoal *Strong in Communication*) and can specify hierarchical decomposition of high-level goals/tasks into lower-level ones (e.g. *Coach team* requires training the team, monitoring its progress and chairing the meetings). More importantly, an SR model also shows how the different intentional elements of the actor contribute to each other to achieve the actor's goals. As an example, being *Strong in Communication* helps achieving the *Facilitated team collaboration* goal. Other kinds of contributions are *Make* (the strongest positive one), *Hurt* and *Break* (the most negative one). These values will be used in the next section to evaluate the feasibility of adopting a software process in a given organization.

### 3.3. Analyzing the risks of adopting an agile process based on its social models

One of the benefits of intentional models is the ability of revealing important social aspects of agile methods that will help organizations trying to adopt them. We have already hinted before the ability of models in this area, for instance, to clearly visualize responsibilities, dependencies, critical roles,... In this section we take a closer look to this topic and focus on the use of our models to analyze the risks of adopting an agile process, in terms of vulnerability points that potentially threaten any adopting organization.

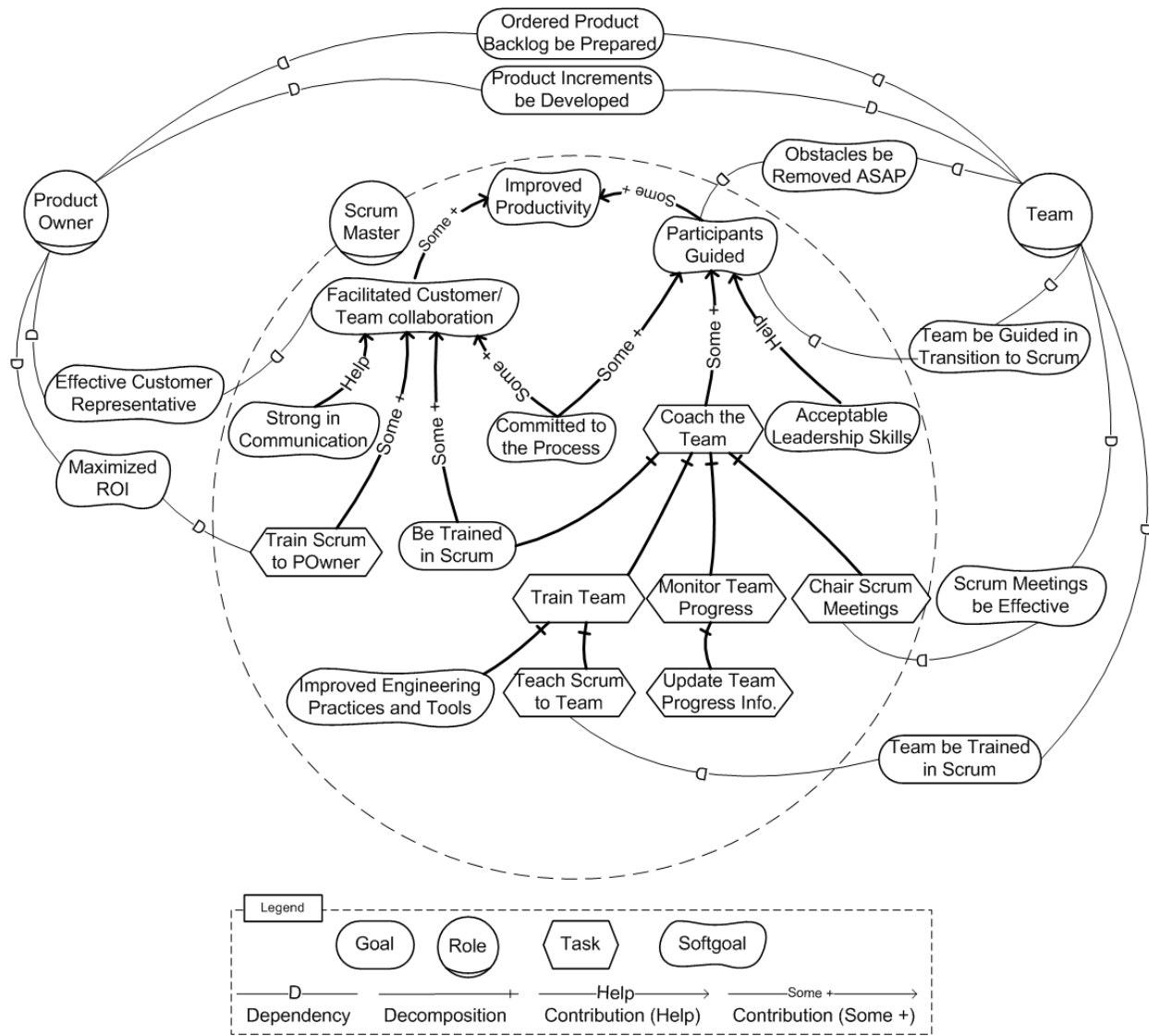


Figure 4. Representing the internal rationales of process actors

Social models can be analyzed to highlight two types of vulnerabilities in agile methods. The first category of vulnerabilities, are those rooted in dependencies and can be identified looking at SD models. Having well-trained and qualified actors for a certain role does not ensure the successful achievement of their method goals, as other actors they rely on may fail. For instance, the SD model of Scrum (Fig. 3) expresses that even a Team of skilled programmers might fail to adopt Scrum if the Scrum Master fails to guide them in their transition to Scrum.

The second category is rooted in the process softgoals and can be identified in SR models. As mentioned earlier, the softgoal notations can refer to social

qualifications, as well as technical expectations. If any of the softgoals fails (which generally deal with human capabilities), the overall process goals might be affected. For instance, if an Scrum Master does not have *Acceptable Leadership Skills* or fails to be *Committed to the Process*, the achievement of its softgoal *Participants Guided* will be threatened, and the problem will be even propagated to the other actors (through dependency relations).

Table 2 summarizes a list of identified vulnerabilities of the Scrum process, based on analysis of SD and SR models. It also represents the corresponding Scrum failure points reported in [17]. Our models helped to predict most of the failure points reported in [17], and

even some new potential vulnerabilities.

Intentional models can be used to systematically analyze the conflicts between the current organizational setting and the method requirements. In other words, they can be used to evaluate the satisfaction (or denial) degree of each method goal, in the presence of specific situational factors. By the end of analysis, and given the list of method goals/requirements the organization is unable to satisfy with the current structure, the organization can decide on the feasibility of adopting the process. The  $i^*$  forward evaluation strategy [20] can be used to propagate and anticipate the effects of situational factors on process goals. The evaluation process starts by tailoring the process model based on the situation of organization/project, and then proceeds by assigning initial values to the leaf-node elements of the SR model, which their achievement degree can be estimated based on situational criteria. After initializing all of the possible leaf nodes, the evaluation process completes by propagating the initial values up to the other model elements. Detailed description of  $i^*$  forward evaluation can be found in [20].

#### 4. Case Study: Adopting Scrum for an undergraduate course project

To validate the benefits of our social perspective on process modeling we applied our approach on an experience of adopting Scrum in a classroom study. The Scrum process was used for the programming project of an undergraduate course in Computer Science. The course had about 160 second-year students, grouped in 40 teams, with 10 teaching assistants (TAs). Each team had to develop the same mobile application for Blackberry devices using Scrum as the development process. The TA in charge of each team played the role of Scrum Master for the team. All TAs and students were being trained in Scrum (during a one-hour lecture), and the lecturer played the role of Product Owner. The first author of this paper was assigned as a head TA, responsible for tailoring, adopting, and monitoring the Scrum process.

The goal of this section is to illustrate 1 - how we tailored Scrum to this specific context by analyzing the constraints of our scenario; 2 - how we analyzed the possible risks of adopting the tailored process, based on the social model of Scrum; and 3 - to what extent our analysis results were substantiated in practice. In a real scenario, the organization should try to find proper solutions for all potential risks, before enacting the process. Due to the constraints of our course-based scenario, fixing all possible risks was not possible. However, for the purpose of this article, this has been

useful to compare the observed problems the teams had when adopting the process with the risks detected a priori with our model-based analysis.

##### 4.1. Tailoring Scrum Process, based on situational constraints

In our scenario, we encountered the following restrictions:

- 1) Most of TAs were not so familiar with mobile application development.
- 2) TAs were accessible only at certain times (their office hours).
- 3) Development cycles were supposed to be exactly three weeks (with possible extension of at most 24 hours).
- 4) Students had little experience in programming.
- 5) Students could not have regular daily meetings.
- 6) Students could not always come together and work at the same time/place.

To adapt the process to these restrictions, we decided a list of actions to be performed before the start of the course. An expert mobile programmer conducted a training session for TAs to become familiar with mobile programming (though this was not enough); the course lecturer asked the TAs to stay in email contact with their teams out of office hours, and set up a bulletin board to improve the communication with students; we reduced the length of sprints from the default 4 weeks to 3; students were supposed to use a wiki page to reflect upon their daily activities, as a replacement for daily Scrum meetings and a solution to the problem of not being always co-located.

Taking into account the restrictions of our classroom setting and the partial corrective actions we could conduct, we then went to specify the social models for our specific enactment of the Scrum process (Figure 5) and proceeded with the analysis of those models to detect those goals of the generic method that we were potentially unable to achieve, which would highlight the risks of our particular development process.

To generate the organization-specific version of the process model we first built the generic social model of Scrum process, and then refined it based on situational constraints and corrective actions. As shown in Figure 5, symbols depicted in red represent our refinements to the social model of Scrum process. We added the softgoals *Tech Knowledgeable* and *Availability* (with their corresponding contributions) to the SR model of Scrum Master; and the softgoal *Programming Experience* and tasks *Use Email*, *Use Bulletin Board*, and *Use Wiki* to the SR model of Team.

Table 2. Identified Scrum vulnerabilities; Source Model: SD Strategic Dependency, SR Strategic Rationale; Corresponding category of Scrum failure reported in [17]

Vulnerability	Source Model	Category
Product Owner Not to be <i>Engaged in Development</i>	SD	Product Owner
Product Owner Fails in <i>Ordered Product Backlog be Prepared</i>	SD	Product Owner
Team Not being <i>Guided in Transition to Scrum</i>	SD	Team
Team Not to have <i>Obstacles Removed ASAP</i>	SD	N/A
Scrum Master Not to be <i>Trained in Scrum</i>	SD	Sprint Planning
Scrum Master Fails to run <i>Effective Scrum Meetings</i>	SD	Daily Meeting
Scrum Master Not to be <i>Strong in Communication</i>	SR	Scrum Master
Scrum Master Not to be <i>Committed to the Process</i>	SR	Scrum Master
Scrum Master Fail to achieve <i>Facilitated Customer/Team Collaboration</i>	SR	N/A
Scrum Master Not to have <i>Acceptable Leadership Skills</i>	SR	Scrum Master
Scrum Master Not respecting <i>Macromanaging</i>	SR	N/A
Scrum Master Fails to keep <i>Participants Guided</i>	SR	N/A
Scrum Master Fails to have <i>Effective Customer Collaboration</i>	SR	N/A
Team Fails in <i>Time-Boxed</i> planning	SR	Sprint Planning
Team Not having <i>Correctly Estimated</i> backlog	SR	Sprint Planning
Team Not to be <i>Self-Managing</i>	SR	Team
Team Not to be <i>Self-Organizing</i>	SR	Team
Team Not to be <i>Cross-functional</i>	SR	Team

#### 4.2. Analyzing the social model of tailored process

Apart from the generic vulnerabilities identified in Table 2, the customization of the Scrum process has also introduced some new possible points of failure that may affect our capacity to effectively adopt the Scrum process. For instance, adding the softgoal *Tech. Knowledge* to the boundary of *Scrum Master*, highlighted the point that Scrum process might fail, if the Scrum Master does not have adequate technical knowledge. Similarly, the poor availability of Scrum Masters can have negative impact on the satisfaction of process goals.

As explained before, for evaluating the satisfaction (or denial) degree of process goals, possible leaf nodes should be initialized with satisfied or denied labels (depending on our particular enactment constraints), and then those values be propagated up to the other model elements. Figure 5 shows the result of initialization and evaluation of tailored Scrum process. For instance, for the subject case study, we initialized the softgoal *Programmers be Experienced* of *Team* to *Partly Denied*, since students had very limited experience of software development.

After propagating the initial values through the SR model, we gained a prediction of how well each method goal can be satisfied (see the resulting labels in root goals in figure 5). For instance, the evaluation results foresaw that the softgoal of *Team* for having *Accurate Sprint Plan* would become *Partly denied*, due to the predictable failure of its dependency to *Scrum Master*, as well as poor evaluation of its contributing

softgoal. As another example, consider one of the main objective of *Scrum Master*, which is the softgoal *Participants Guided*. Even if we tried to improve the availability of TAs by adding email support, which favors the *Committed to the process* softgoal, the limitations in the technical domain knowledge could result in weak denial of the softgoal *Participants Guided*.

Social models of agile processes can also be analyzed with respect to their common problems. For instance, one of the typical conflicts with what agile methods require is the poor participation of customers in the development process. As shown in Figure 5, this limitation would deny the goal of *Product Owner* to be *Engaged in Development*, which, in its turn, endangers the softgoal of *Scrum Master* for *Facilitated Customer/Team Collaboration*. Consequently, this issue causes the *Scrum Master* to fail at satisfying its own softgoal of *Improved Productivity*, and the dependency of *Team* for having *Obstacles Removed*. This chain of negative impacts can be further analyzed within the SR model of *Team*.

#### 4.3. Credibility of analysis results

To test the credibility of this kind of analysis, we compared the results of the systematic (model-based) evaluation of process goals, with what the process head-TA observed regarding the average results of each group. In most cases, the observed values of process goals were the same as model-based ones. For example, we observed that student teams were somehow successful in being *Self Managing*, which

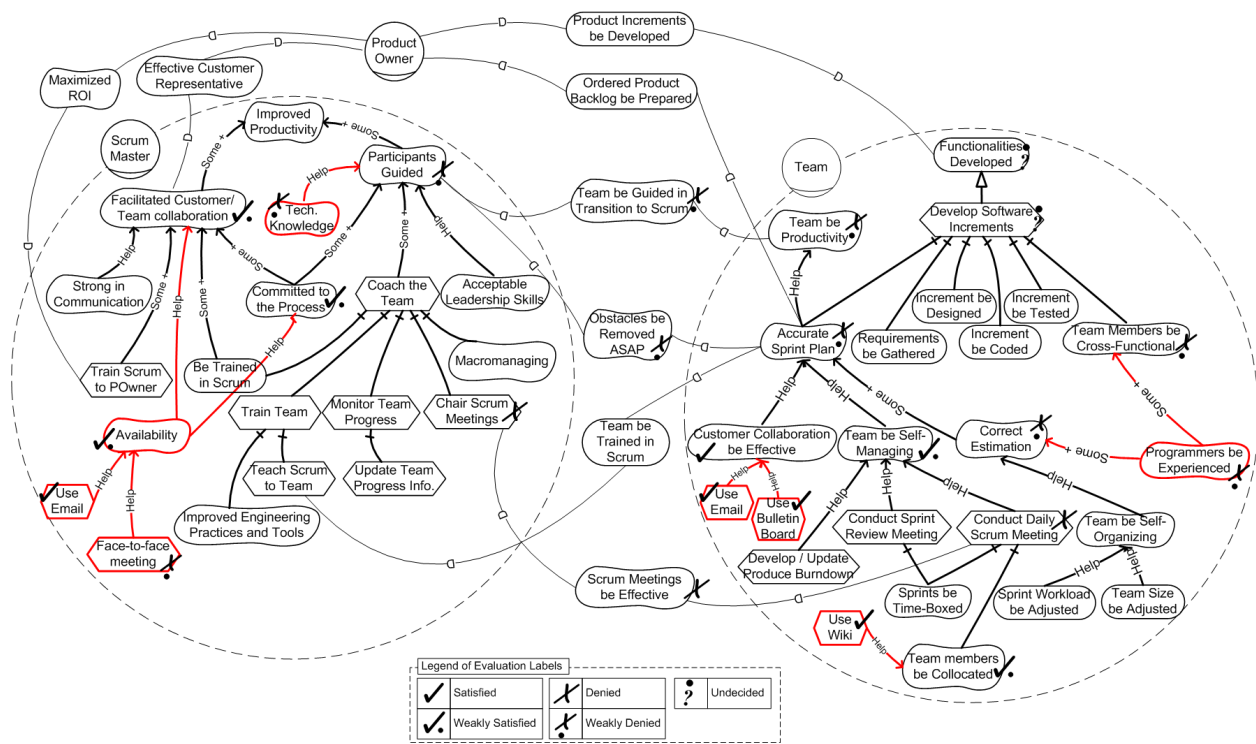


Figure 5. Tailored Scrum model for our classroom scenario

was in accordance with the model-based analysis results. However, there were some cases of deviation as well. For instance, based on our model evaluation we expected that the softgoal (quality attribute) of team members in being *Cross Functional* would be *Weakly Denied*. However, we observed that this quality of team members gradually improved, as they were approaching final iterations. Therefore, at the end of project, we evaluated this softgoal as *Weakly Satisfied*. Table 3 summarizes the results and justifies the given values.

If we take a closer look at the comparison (Figure 6), we see for 6 out of 10 process goals, the results of systematic evaluation was the same as the observed values. This analysis also shows that the framework tends to evaluate process goals conservatively, since it made (in this case) 4 out of 10 under-estimations (observed value better than model-based value), yet making no over-estimation. however, It should be mentioned that in 3 out of 4 under-estimated cases, the model-based values were *Weakly Denied*, whereas the observed values were *Weakly Satisfied*. A possible justification to this slight deviation could be the lack of strongly distinctive definition for these two values, regarding the qualitative nature of the *i\** framework.

		Model-based Analyzed Value						
		D	WD	UD	WS	S	Legend	
Observed Value	D	0	0	0	0	0		Over-Estimated
	WD	0	2	0	0	0		Under-Estimated
	UD	0	0	0	0	0	D	Denied
	WS	3	0	0	3	0	WD	Weakly Denied
	S	0	0	1	0	1	UD	Undecided
						WS	Weakly Satisfied	
						S	Satisfied	

Figure 6. Analysis of the expected vs obtained results

## 5. Related Work

As mentioned before, most of the current process modeling approaches are activity-oriented, as they focus on ordering of process activities and products. Weerd et al. in [6] introduced Process-Data Diagram (PDD), which is a process modeling technique, developed by integrating UML Activity Diagrams and Class Diagrams. Although PDD supports the definition of process roles, it has no feature to represent qualifications expected from roles, or their objectives and social dependencies. The social aspects of software

Table 3. Analyzed vs. Observed values of Scrum method (soft)goals, applied in the case study. (WS: Weakly Satisfied; S: Satisfied; UD: Undecided; WD: Weakly Denied; D: Denied)

Actor	Goal/Softgoal	Expected	Observed	Description
Team member	Accurate Planning	WD	WS	Earlier Sprint plans were faulty but their planning capability improved, as they became more familiar with technology and needs
Team member	Collocated	WS	WS	Wiki-based communication helped teams to overcome the problem of not being always collocated
Team member	Productivity Improved	WD	WS	Productivity improved during the last iterations
Team member	Functionalities Developed	UD	S	Our analysis did not reach any conclusive result for this goal but teams ultimately developed the functionalities
Team member	Effective Customer Collaboration	S	S	Email contact and bulletin board were enough for this project
Team member	Cross-Functional	WD	WS	During the semester, team learned different topics in programming, testing, and designing???
Team member	Correct Estimation	WD	WD	Teams usually fall into under or over estimation
Team member	Self-Managing	WS	WS	From the beginning, students tried to divide the work, however it was not always adjusted
Scrum master	Participants Guided	WD	WD	Due the incomplete tech knowledge, TAs could not guide the students by rapidly removing their obstacles.
Scrum master	Facilitated Customer/Team Collaboration	WS	S	The bulletin board plus the rapid response to emails proved to be enough
Scrum master	Committed to the Process	WS	WS	TAs tried to be responsive to the teams

processes is also overlooked in the standard process engineering metamodel (SPEM 2.0) [13], as it does not deal with interactions among team members, or their contribution to the process goals.

Goal-oriented process modeling has been widely advocated for business process modeling [21], [22]. This modeling approach has been also proposed for software processes [23], [24], [30], [31]. Lee in [23] described a systematic method for process redesign called Goal-based Process Analysis (GPA). This method takes advantage of goal hierarchies to identify missing objectives, and propose alternatives for reaching a goal. Leijen and Baets in [24] proposed a framework for reengineering knowledge-intensive processes in which the organization is modeled as a goal-directed planner for exhibiting proper response in reaction to a stimulus. However, neither of these techniques is concerned with the representation and analysis of social aspects of software/agile processes nor on the analysis of their vulnerabilities and key success or failure elements. [30], [31] are proposed to take advantage of method rationales for improving the process of method engineering. Rolland and Gorsz in [33] proposed an intention-based language for representing software processes, in which process fragments were modeled as pairs of (*situation, decision*). However, the focus of that work was on assisting process designers with situational solutions, rather than modeling social aspects of software processes.

On the other hand research on social aspects of agile methods studies the effect of social factors such as knowledge sharing, motivation and customer collaboration on agile practices [25], the social issues that

hinder the process of introducing an agile method to an organization [1], [2], [4], [5] and the impact of cultural factors on software development [26]. Our approach allows an explicit representation of all these social factors, which can be analyzed and incorporated into the decision-making process of an organization when selecting a software process for a given project. This paper goes beyond the earlier use of *i\** for process modeling [19], by focusing on agile methods and the prescriptive presentation of their social aspects.

## 6. Conclusions and further work

A complete understanding of a software process is hard to achieve by having a single model of the process [7]. This paper proposed a goal-oriented approach for modeling social aspects of agile methods to complements other more activity-oriented views. It focuses on the representation of the relationships and dependencies between the roles involved in the process, the responsibilities of each role, and the skills needed to play each role. It also aims at analyzing several kinds of possible vulnerabilities of the method that may endanger its successful introduction in an organization. This view can also be helpful in more traditional software processes.

With our approach, organizations may visualize and evaluate how well the different agile methods may fit with their existing internal structure and minimize the possible risks derived from their adoption. Once decided to implement a given method, our approach also provides a common understanding of the method to all different actors: what are their responsibilities,

what are the consequences of not doing their job, and how to collaborate to fulfill their goals.

As a further work, we plan to first develop a library of goal models for agile processes to facilitate the evaluation and comparison of different agile methods from a social perspective. We are also interested in formalizing the method for adapting a given software process to the reality of each organization by matching the organization structure to the process goal-model template. This affects the development of software project management tools that must take into account how a process is really applied within an organization in order to provide a customized support for the process. Finally, we would like to extend the result of our work to the field of Method Engineering where software processes are built by combining several method fragments [27]. We believe that understanding the social aspects of method fragments can help method engineers in assessing the vulnerabilities of new processes.

Of course, tailoring a method is still a manual task and the quality of the results may depend a lot on the ability of the method expert to identify and represent in the social model and the constraints of each specific organization. However, we are currently in the process of developing an evidential repository of method fragments to support the the credibility of process social models. This repository is based on empirical studies and contains the situational evidences of enacting method fragments in different situations. Besides, we are working to develop a customized *i\** forward evaluation procedure, which considers some of the specific concerns that arise during the analysis of software process models.

## References

- [1] Cohn, M., Ford, D.: Introducing an Agile Process to an Organization. *Computer*, vol. 36, No. 6 pp. 74-78 (2003)
- [2] Boehm, B., Turner, R.: Management challenges to implementing agile processes in traditional development organizations. In: *Software*, IEEE, vol. 22, No. 5, pp. 30-39 (2005)
- [3] Osterweil, L.: Unifying Microprocess and Macroprocess Research. *SPW 2005*, Vol. 3840, pp. 68-74. Springer, Heidelberg (2005)
- [4] Harald, S., Martin, H.: Introducing an Agile Process in a Software Maintenance and Evolution Organization. In: *Proceedings of the Ninth European Conference on Software Maintenance and Reengineering*, pp. 256-264. IEEE Computer Society (2005)
- [5] Abdelnour-Nocera, J., Sharp H.: Adopting Agile in a Large Organisation. In: *Lecture Notes in Business Information Processing*, vol. 9, pp. 42 – 52, Springer, Heidelberg (2008)
- [6] O'Donnell, M.J., Richardson, I.: Problems Encountered When Implementing Agile Methods in a Very Small Company. In: *Software Process Improvement*. Vol. 16, pp. 13-24, Springer Berlin Heidelberg (2008)
- [7] Derniame, J.C., Kaba, B.A., Wastell, D.G. (eds.): *Software Process: Principles, Methodology, Technology*. Springer-Verlag (1999)
- [8] Agile Manifesto (2001). Manifesto for agile software development, <http://www.agilemanifesto.org>, visited at December (2008)
- [9] Schwaber, K., Beedle, M.: *Agile Software Development with Scrum*. Prentice Hall PTR (2001)
- [10] Boehm, B., Turner, R.: *Balancing Agility and Discipline, a Guide for the Perplexed*. Addison-Wesley (2003)
- [11] Colette, R.: A Comprehensive View of Process Engineering. In: *Proceedings of the 10th International Conference on Advanced Information Systems Engineering*, LNCS, pp. 1-24, Springer-Verlag (1998)
- [12] Soderstrom, E., Andersson, B., Johannesson, P., Perjons, E., Wangler, B.: Towards a Framework for Comparing Process Modelling Languages. In: *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, pp. 600-611, Springer-Verlag (2002)
- [13] SPEM 2.0: Software Process Engineering Metamodel, Version 2.0. Object Management Group (OMG) (2008)
- [14] Kolcz, K.: Using SPEM/UML profile to specification of IS development processes. Master Thesis, School of Software Engineering, Blekinge Institute of Technology, Sweden (2006)
- [15] Abrahamsson, P., Warsta, J., Siponen, M., T. , Ronkainen, J.: New directions on agile methods: a comparative analysis. In: *Proceedings of the 25th International Conference on Software Engineering*, pp. 244-254, IEEE Computer Society (2003)
- [16] Cockburn, A., Highsmith, J.: Agile Software Development: The People Factor. In: *Computer*, vol. 34, No. 11, pp. 131-133 (2001)
- [17] Sutherland, J.: Seven ways to fail with scrum. In: *The Journal of Defense Software Engineering*, vol. 20, No. 4, pp. (2007)
- [18] Yu, E., Mylopoulos, J.: Understanding “Why” in Software Process Modelling, Analysis, and Design. *Proceedings of 16th International Conference on Software Engineering*, Sorrento, Italy, pp. 159-168, (1994)

- [19] Yu, E.: Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering, pp. 226-235, IEEE Computer Society (1997)
- [20] Horkoff, J., Yu, E.: A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models. In Proceedings of CEUR Workshop in CAiSE,09 (2009)
- [21] Lapouchnian, A., Yu, Y., Mylopoulos, J.: Requirements-Driven Design and Configuration Management of Business Processes, In Proc. 5th International Conference on Business Process Management (BPM 2007), LNCS Vol. 4714, pp. 246-261, Springer-Verlag, Berlin Heidelberg (2007)
- [22] Greenwood, D., Rimassa, G.: Autonomic Goal-Oriented Business Process Management. In: proceedings of third International Conference on Autonomic and Autonomous Systems (2007)
- [23] Lee, J.: Goal-Based Process Analysis: A Method for Systematic Process Redesign. In: Proceedings of the conference on Organizational computing systems, pp. 196 - 201, ACM, New York (1993)
- [24] Hans van, L., Walter, R.J.B.: A Cognitive Framework for Reengineering Knowledge-Intensive Processes. Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03). IEEE Computer Society (2003)
- [25] Michael, J., Frank, M., Bj, rnar, T.: Human and social factors of software engineering: workshop summary. In: SIGSOFT Software Engineering Notes, vol. 30, No. 4, pp. 1-6 (2005)
- [26] Sridhar, N., RadhaKanta, M., George, M.: Challenges of migrating to agile methodologies. In: Commun. ACM, vol. 48, No. 5, pp. 72-78 (2005)
- [27] Weerd, I.v.d., Brinkkemper, S., Souer, J., Versendaal, J.: A situational implementation method for web-based content management system-applications: method engineering and validation in practice. In: Software Process: Improvement and Practice, vol. 11, No. 5, pp. 521-538 (2006)
- [28] UML 2.0: Unified Modeling Language, Superstructure Specification. Object Management Group (OMG) (2003)
- [29] BPMN: Business Process Modeling Notation. Object Management Group (OMG) (2006)
- [30] Agerfalk, P. J., Fitzgerald, B.: Methods as Action Knowledge: Exploring the Concept of Method Rationale in Method Construction, Tailoring and Use. 10th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (2005)
- [31] Rossi, M., Tolvanen, J. P., Ramesh, B., Lyytinen, K., Kaipala, J.: Method rationale in method engineering. Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (2000)
- [32] Elahi, G., Yu, E. . A Goal Oriented Approach for Modeling and Analyzing Security Trade-Offs. (Ed.).In: Conceptual Modeling - ER pp. 375-390, (2008)
- [33] Rolland, C., Grosz, G. A general framework for describing the requirements engineering process. IEEE International Conference on Systems, Man, and Cybernetics, 'Humans, Information and Technology'. pp. 818-823 (1994)