Using i* with Scrum: An Initial Proposal

Leonardo Berbare de Araujo, Fábio Levy Siqueira

Escola Politécnica da Universidade de São Paulo, São Paulo, Brazil
leo.berbare@usp.br; levy.siqueira@usp.br

Abstract. Although goal modeling using i* provides several benefits to requirements analysis, it may be difficult to use it with agile methods. This paper proposes a method that combines i* with Scrum, integrating the initial phases of Tropos with Scrum practices. In summary, goals in a Strategic Rationale model are prioritized to a release and then refined into user stories. The stories are implemented in Sprints, following the Scrum activities. This paper also reports the use of this method to develop a mobile app that brings political decisions closer to the Brazilian electorate.

Keywords: i-star, tropos, requirements, scrum, agile, user story, planning

1 Introduction

User story [1] is a popular requirements representation used in agile software development projects. Even though its textual template represents the goal behind a feature (in the "so that" part), it is not possible to reason about goals, especially higher-level goals, or reason about how goals are refined into requirements. This information, for example, is important when creating user stories, or when a product owner selects the features to be implemented in an iteration. Even if some agile methods, such as Scrum [2], do not impose using user stories, it is not clear how to use goal modeling in agile projects, with changing requirements and continuous delivery of software.

This paper proposes a method that combines goal modeling, using i* models, with agile software development. We integrate the initial phases of Tropos, Early Requirements and Late Requirements, with Scrum practices. Differently from works that propose transforming i* models into user stories [3], transforming user stories into i* models [4], or obtaining a goal net model from user stories [5], our method involves refining a partial i* model into user stories that will be used to guide the software development.

To present the method, this paper is organized as follows: Section 2 presents Scrum and user story. Section 3 describes the method, and Section 4 presents an experience report of how the method was used to develop a mobile app. Finally, Section 5 presents a discussion and some conclusions.

2 Scrum and User Story

Scrum is an agile framework for managing the development of complex products and services [2]. A Scrum team is composed of three roles: a product owner (who decides which features will be developed and their priority), a Scrum master (who acts like a coach), and a development team (a cross-functional and self-organized group of developers).

A project using Scrum begins with the creation of a product backlog, representing a list of requirements for the system. This backlog is created and prioritized by the product owner, while the development team helps estimating each item in it. The work on items of the product backlog is executed in sprints. A sprint is a time-boxed iteration to create a product increment with value to the customer or user. Based on the product backlog and the vision of the product owner, the Scrum team agrees on a sprint goal, which may be a specific set of backlog items or a set of features. The items of the product backlog that will be developed in the sprint are organized in a sprint backlog, along with a plan to deliver the product increment. To create this plan, Scrum teams normally break each item into tasks, and estimate the effort to complete each one of them [2]. The framework does not describe how tasks should be implemented, but it describes some important events. A brief daily meeting should be performed by the development team and the Scrum master to understand how the development is progressing and what issues should be addressed. The other two events should be conducted at the end of the sprint: the sprint review and the sprint retrospective. The first is a review of product created in the increment, while the second is a review of the process, focusing on process improvement.

While Scrum does not impose a requirement representation for the product backlog, many teams use user stories [2]. According to Cohn, a user story “describes functionality that will be valuable to either a user or purchaser of a system or software" [1]. Differently from other requirements representations, a user story is not a detailed specification; it is a reminder for a conversation between the development team and the stakeholders [1].

3 Method

In order to bring goal modeling into the Scrum framework, we propose a method based on Tropos [6]. The method comprises five phases: Early Requirements, Initial Late Requirements, Goal Refinement, Sprint Planning, and Implementation. Unlike Tropos, the method does not impose an Agent-Oriented Software Engineering approach.

The first phase, Early Requirements, is similar to the one proposed in Tropos, but adapted to the Scrum framework. Therefore, a developer with requirements engineering skills and the product owner model the system's stakeholders into social actors: defining roles and intentions. They create the Strategic Dependencies (SD) Model by listing the possible dependencies between each pair of social actors. Then, they make the Strategic Rationale (SR) Model by listing possible hardgoals, softgoals, plans (we use "plan", from Tropos, instead of "task" to avoid a confusion with Scrum's "task"), and resources within each social actor's boundary and describing how these goals affect each other.

On Initial Late Requirements, the second phase, the developer and product owner introduce the system-to-be as a new social actor, and update the SD Model to include dependencies the initial actors may have with the system (which sometimes replace old dependencies). Similar to the first phase, they make the SR Model for the system defining goals that help accomplish the dependencies in which the system is a dependee.

The third phase, Goals Refinement, is the core of the method. This phase should be executed as a release planning [2], considering several sprints. The product owner sorts the list of dependencies with the system according to their priority. He or she selects the ones essential to the next release of the system and, with support of the developer with requirements engineering skills, makes a list of the system's internal goals that help accomplish these dependencies. They sort goals based on their importance and the product owner then selects which ones to refine. Refinement of a goal consists of: describing the goal in a sentence or two; decomposing it into plans that serve as means to the goal (giving a title to each plan and describing how they compose the goal); and giving each plan a set of user stories that should cover it. The development team should estimate each user story, creating a product backlog.

The fourth and fifth phases, respectively, Sprint Planning and Implementation, is executed in all sprints. Sprint Planning consists of the product owner selecting a subset of user stories (therefore, a set of plans and goals) with a theme in mind and considering the team velocity. The team should split each item into tasks and create a sprint backlog.

As in Scrum, an Implementation phase follows each Sprint Planning phase, and implementation methods should vary according to team and project. Yet, the phase starts by the assignment of developers to the sprint backlog items. The progress of development should be available to the team by pointing out which items (plans, user stories, and their tasks or components) have been completed.

4 Applying the Method

The method was created and used during a capstone project of a Computer Engineering course at Universidade de São Paulo (USP). The idea was to develop a mobile app, named Appopuli, which brings political decisions closer to the Brazilian electorate. The team was composed of two final-year students, working as developers and product owners, and an advisor, working as a Scrum master.

During the Early Requirements phase, we identified the following actors: Electors, Politicians, Political Parties, and Press. We elicited a set of dependencies such as “an Elector depends on the Press to find news about candidates” and then the rationale for each actor, writing goals like “recognize good candidates” (an Elector goal).

During the Initial Late Requirements phase, we included the System actor and elicited dependencies such as “the Elector depends on the System to express their opinion.” The rationale of the System actor resulted not only in many hardgoals representing subsystems and components, but also in a few softgoals representing desirable functioning qualities.

In the Goals Refinement phase, we decided that dependencies with the Political Parties and the Press had lower priorities than the Electors’ and Politicians’. We also narrowed the scope of our project to the Municipal Chamber of São Paulo. Therefore, only 15 hardgoals out of 40 goals inside the System actor boundary were refined in order to create a minimum viable product during the sprints. This phase resulted in 30 plans such as “Rate a politician between 1 and 5 stars,” which has a small set of user stories, includ-

ing “an Elector may change the number of stars previously given to a politician.” An example of the refinement is presented in Fig. 1.

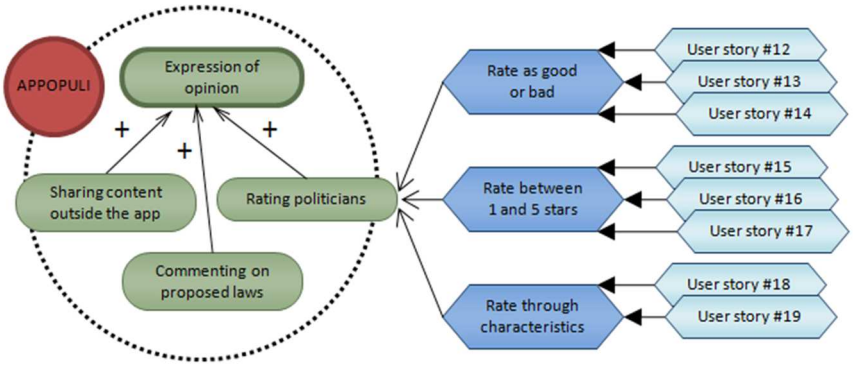

Fig. 1. Refinement of the goal “Rating politicians” into plans and user stories.

The rest of the development was divided in four sprints. Each sprint executed both the phases Sprint Planning and Implementation. Only one of the final-year students executed these phases, as the other student left for an exchange period. The first sprint focused on building a simple app that shows basic info on every alderman of São Paulo. Planning it meant choosing which user stories would be implemented. For instance, picking “a user may search aldermen by name” rather than “a user may search aldermen by political party.” Differently from the method, we split the user stories into tasks during the Implementation phase. The three other sprints were planned similarly.

During Implementation, each sprint backlog was managed through the list of tasks to satisfy a set of user stories. Some tasks were described as the classes and methods to be implemented. The App was developed for Android phones, using Java; the server side of the application was developed in Ruby, using the framework Ruby on Rails and a PostgreSQL database. By the end of the fourth sprint, users could: sign up at the Android app; choose aldermen and proposed laws to follow; see any activity from followed concerns on a timeline; check details of a proposed law (including their original PDF documents); among other functionalities.

5 Discussion and Conclusion

This paper proposes a method that combines goal modeling, using i* and based on Tropos, with the agile framework Scrum. We also describe the use

of this method to develop a mobile app. The proposed method is based on the idea that goals are more stable than requirements. Therefore, the goal model will not change frequently, and it would be possible to use it in release planning with few changes during sprints. Yet, the benefits of goal modeling would allow a better understanding of the system to be built, help reasoning about alternatives, and prioritize the development based on goals. These benefits should compensate the addition of an artifact – which is a disadvantage from an agile perspective.

As this is an initial proposal, there are some important future works. The method should be improved by including some guidelines and a more detailed description, specially in the Goal Refinement phase. In addition, it should be evaluated in a project with a real product owner and a bigger development team. Finally, the benefits of the method should be analyzed considering its impact to agility.

Acknowledgements. We thank Diego Henrique dos Reis Marques for his participation in the project.

References

1. Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley Professional, Boston (2004).
2. Kenneth S. Rubin: Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley Professional, Upper Saddle River, NJ (2012).
3. Agra, C., Sousa, A., Melo, J., Lucena, M., Alencar, F.: Specifying guidelines to transform i* Model into User Stories: an overview. In: Proceedings of the Eighth International i* Workshop. pp. 109–114. CEUR, Canada (2015).
4. Jaqueira, A., Lucena, M., Alencar, F.M., Castro, J., Aranha, E.: Using i* Models to Enrich User Stories. Presented at the i* Workshop, Valencia (2013).
5. Lin, J., Yu, H., Shen, Z., Miao, C.: Using goal net to model user stories in agile software development. In: 2014 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD). pp. 1–6 (2014).
6. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. Auton. Agents Multi-Agent Syst. 8, 203–236 (2004).