Requirements Engineering with Agent-Oriented Models

Tanel Tenso and Kuldar Taveter

Department of Informatics, Tallinn University of Technology, Raja 15, 12618, Tallinn, Harjumaa, Estonia

Keywords: Requirements Engineering, Agent-Oriented Modeling, Systems Modeling, User Stories, Agile Development.

Abstract: This paper provides a solution how to gather requirements using modified principles of agent-oriented modeling. By simplifying the concepts of agent-oriented modeling for requirements engineering we have found that this kind of modeling can be used in various software development projects including agile software development for agile requirements engineering. To verify our theories, we've used our practice in two real-life projects. This paper contains summary of ideas and practices developed during these projects.

1 INTRODUCTION

There is evidence that main cause for the failure of software projects is missing stakeholder input and invalid requirements (Group, 1995; El Emam and Koru, 2008). We have encountered these problems also in our work. This has prompted us to come up with a novel idea to mitigate the problem of collecting and documenting requirements. In the center of our approach to requirements engineering (RE) lies using the goal modeling technique from agent-oriented modeling (AOM) as a backbone for requirements elicitation and representation. Goal modeling alone is not a new idea because goal-based RE is described by several authors (Hull et al., 2011; Sillitti and Succi, 2005; Dardenne et al., 1993; van Lamsweerde, 2001; van Lamsweerde, 2009). The novelty of our approach is linking goal models to user stories (Cohn, 2004), which are regarded as most widely used requirements documentation artifacts in agile software development (ASD) (Ramesh et al., 2010; Cao and Ramesh, 2008). In this paper we present our technique for improving RE in both agile and non-agile settings.

We have successfully tried out our approach in two real-life projects. The nature of these projects is different: one is concerned with complementing an existing client-server web application with a new functionality, while the other addresses RE for a large crisis simulation system. In this paper we use the examples from these experiments for illustrating our approach. We plan to publish findings from these applications as detailed case studies.

The structure of this paper is as follows. In Section 2, we provide short overviews of RE, ASD and briefly describe AOM. In Section 3, we present our approach of using AOM in RE, and illustrate it with examples from application projects. In Section 4, we discuss anecdotal evidence received from real-life projects and set future research directions.

2 BACKGROUND

To summarize RE very shortly we can say that it is process for finding out, analyzing, documenting and checking requirements for what and how should some complex system work (Ian Sommerville and Sawyer, 1997; Hull et al., 2011; Kotonya and Sommerville, 1998; Sommerville, 2010). RE is quite a wide area and in the context of this paper we concentrate on the requirements elicitation activity. We leave validation and management of requirements out of the scope.

ASD can be viewed as consisting of agile practices. In ASD the emphasis is on code and collaboration rather than on documentation and up-front elicitation (Alliance, 2013; Paetsch et al., 2003). In principle RE processes are present in ASD but they are less distinctive and iterative. Identifying a link between RE and ASD has been the goal for several researchers (Haugset and Stalhane, 2012; Paetsch et al., 2003; Cao and Ramesh, 2008).

Due to preferring collaboration over documentation, there are different approaches for documentation in agile projects. Some are quite radical by minimizing documentation to almost non-existing (Cockburn, 2002), while some others acknowledge the need for some documentation (Ambler, 2002). We take the latter viewpoint. Several researchers (Ambler, 2002; Beck, 2008; Kazyrevich, 2010) have proposed ideas to be considered when collecting and documenting requirements in ASD projects. We have used principles that have been expressed in a more detailed context by (Ambler, 2002). The author subsumed his research results under the term ”Agile Modeling” (AM), which is essentially a practice-based methodology for effective modeling and documentation of software-based systems. For example, we have adapted the number and content of models usually used in AOM to match the AM principle “just barely enough”.

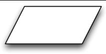
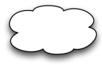
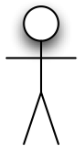
User story is one of the most popular requirements artifact in ASD (Cohn, 2004; Ramesh et al., 2010; Paetsch et al., 2003). There are several approaches to user stories in agile development, out of which we have employed in our approach a format proposed by (Cohn, 2004). A user story includes a written sentence or two and should invoke a series of conversations about the desired functionality. The format of composing user stories can be described as follows:

1. A user story is a short, simple description of a feature described from the perspective of the person or role who desires the new capability, usually a user or customer of the system.
2. The template for user stories could correspond to the following pattern according to (Cohn, 2004): As a ⟨type of user or role⟩, I want/must ⟨some goal⟩ so that ⟨some reason⟩.
3. A user story must be small enough to be implemented within one iteration. Large user stories must be divided into smaller user stories.

Final cornerstone of our approach is agent-oriented modeling (AOM) (Sterling and Taveter, 2009). AOM is a holistic approach for analyzing and designing socio-technical systems consisting of humans and technical components. The case studies (Miller et al., 2011; Miller et al., 2012; Taveter et al., 2012) and several others have demonstrated that the agent-oriented paradigm is useful for requirements engineering and design, irrelevant of whether the eventuating system is a multi-agent system in the classical sense (Wooldridge, 2001). AOM is centered on the notion of agent[1].

We have selected AOM as the central part of our RE and design approach because the types of models included by AOM are clear and understandable for all stakeholders when building any complex socio-technical system (Miller et al., 2011). AOM canonical models are described in detail by (Sterling and Taveter, 2009). Next we will give an overview of two models that we decided to use from the AOM methodology in our approach for RE. These are goal models and behavioral scenarios.

Table 1: Notation for modeling goals and roles.

Symbol	Meaning
	Goal
	Quality goal
	Role
―――――	Relationship between goals
---------	Relationship between goals and quality goals

Goal model can be considered as a container of three components: goals, quality goals, and roles (Sterling and Taveter, 2009). A goal is a representation of a functional requirement of the socio-technical system. A quality goal, as its name implies, is a non-functional or quality requirement of the system. Goals and quality goals can be further decomposed into smaller related sub-goals and sub-quality goals. The hierarchical structure is to show that the subcomponent is an aspect of the top-level component. Functional goals represent functional requirements, while quality goals represent non-functional requirements. Goal models also determine roles that are capacities or positions that are needed to achieve the goals. In the original AOM methodology roles are modeled in detail by role models in the viewpoint of interaction analysis. The notation for representing goals and roles is shown in Table 1.

An example of a goal model is represented on Figure 1. The example is taken from the Release Management System (RMS) project that is briefly described in Section 3. The model expresses that the highest-level goal – purpose – of the system is “Manage Release Lifecycle”. This goal is elaborated into two sub-goals. The first of them – Manage Release Vehicles – is performed by the roles Release Manager and Release Admin. Achieving of the second sub-goal – Manage Product Features – requires the roles Analyst, Architect, Product Manager, and Project Manager. We can also see that Figure 1 represents several quality goals describing non-functional requirements that characterize how functional goals should be achieved. For example, the quality goal attached to the functional goal “Manage Product Fea-

[1]Agent is an entity that performs specific activities in an environment of which it is aware and can respond to changes in the environment.

tures" expresses that Product Features (whatever they are, this is not important here) should be managed in such a manner that they have sufficient information for development.

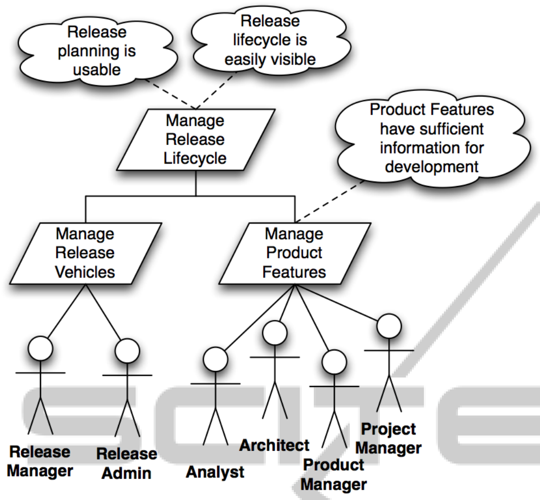


Figure 1: An excerpt of the main goal model for the RMS.

The setup and usage of goal models is quite similar to the principles suggested by (Hull et al., 2011). Like in some other modeling frameworks like i*(Yu, 1996; Bresciani et al., 2004), in AOM non-functional requirements can be represented as quality goals visually separated from functional goals but at the same time linked to them. A resulting goal model contains sufficient amount of information, but compared to i*(Yu, 1996; Bresciani et al., 2004) is still simple to understand for all participants in RE process (Sterling and Taveter, 2009; Miller et al., 2011). We consider goal models to be used for high-level problem domain analysis rather than for system-specific design. We employ for design user stories from ASD methodology.

A behavioral scenario in the AOM methodology describes how agents achieve the goals set for the system by performing a sequence of activities (Sterling and Taveter, 2009). In our approach to RE, we modified the original format of behavioral scenarios. A behavioral scenario of the resulting kind describes the sequence of activities required for achieving the goals, as well as the resources used by and the roles involved in the scenario.

An example behavioral scenario is represented in Table 2. This behavioral scenario consists of a sequence of activities that are required for achieving the "Manage Release Vehicles" goal modeled in Figure 1. The scenario also includes its initiating role(s), triggering condition, and failure condition. The latter describes the state of affairs in case the scenario fails. For each activity is shown the condition for performing the activity. The implicit performing condition is "Sequential".

3 METHOD

Goal based requirements modeling in RE has been found to be useful by several authors (Hull et al., 2011; Sillitti and Succi, 2005). User stories have been connected to RE in several publications about requirements‘ collecting problems in ASD (Sillitti and Succi, 2005; Ramesh et al., 2010; Haugset and Stalhane, 2012). One can conclude based on these sources that user stories can basically be viewed as goals (Haugset and Stalhane, 2012; Vanhanen et al., 2009). This was one important factor contributing to the idea of connecting user stores to AOM goal models.

We have tried our combined approach out in two real-life applications. The first application was developing a RMS. The purpose of the project was to enhance an existing Issue Management System (IMS) with additional functionality for release management. The second was a real-life project on developing a crisis simulation system. The purpose of this project was to provide a universal tool for studying evolvements of different kinds of crises, such as earthquakes, floods, industrial accidents, etc., and training personnel for the crises. Both projects are quite different starting from the scope and ending with the number of participants.

Our purpose in both projects was to find if our approach combining AOM and user stories could improve communication between participants in software process. The rationale for choosing AOM was its simplicity and the fact that goal-based approaches to requirements engineering have considerably improved communication in various software engineering projects (Miller et al., 2011). We chose user stories because they are common for ASD (Cohn, 2004).

In the following, we will outline the software engineering process of our approach. The following description is illustrated by Figure 2:

1. Create the top-level hierarchical goal model:

1.1. Determine the purpose of the socio-technical system being designed. Represent the purpose of the system as the root goal. For example in Figure 1 the root goal is "Manage Release Lifecycle".

1.2. Elaborate the main goal into sub-goals, each representing an aspect of achieving the main goal. For example, in Figure 1 the main goal has been elaborated into the Manage Release

Table 2: Manage Release Vehicles behavioral scenario.

SCENARIO 2					
Goal		Manage Release Vehicles			
Initiator		Release Manager, Release Admin			
Trigger		New planning session begins			
Failure		No Release Vehicle will be produced			
Cond.	Step	Activity	Roles	Resources/ Knowledge Items	Additional comments on design
Optional, Interleaved	1	Create Release Category	Release Admin	Release Category	Define category name and order
	2	Create Release Track	Release Admin	Release Track	Define track name and order
	3	Create Release Status	Release Admin	Release Status	Define status name and order
Sequential	4	Create Release Vehicle	Release Manager	Release Vehicle, Release Track, Release Status, Release Category	Release Vehicle has some preliminary metadata defined
	5	Edit Release Vehicle	Release Admin	Release Vehicle, Release Track, Release Status, Release Category	Release Vehicle metadata is changed to appropriate values when release scope and responsible persons have approved the change
Optional	6	Delete Release Vehicle (Scenario 7)	Release Manager	Release Vehicle, Issue	Release Vehicle is deleted, a connected objects of the Issue type are associated with another Release Vehicle or the connection is deleted

Vehicles and Manage Product Features sub-goals.

1.3. For the main goal and its sub-goals: where appropriate, complement a functional goal with a quality goal, representing a quality aspect of achieving the functional goal. For example, in Figure 1 the functional goal Manage Product Features has been modified by the quality goal Product Features have sufficient information for development".

2. Elaborate the top-level goal model into lower-level goals:

2.1. Choose each sub-goal of the top-level goal model as the main goal.

2.2. Apply step 1 of the process to the main goal.

3. Repeat elaboration of the goal model until you have reached the lowest level of reasonable and achievable goals:

3.1. Lowest level goal model is usually a goal that can be accomplished by single role or accomplishing goal can be easily described.

3.2. Revise goal models at any point whenever more accurate information becomes known.

4. Elaborate lowest level of sub-goals with behavioral scenarios, the format of behavioral scenarios is defined by Table 2:

4.1. Add temporal and conditional perspectives to achieving the goals in behavioral scenarios.

4.2. List goals as activities in behavioral scenarios, adding additional information about what to do

4.3. Specify roles, resources, and knowledge, and any additional information relevant for achieving the goals.

4.4. Behavioral scenario is finished when the achievement of all goals in the goal model is described

5. Link User Stories to Behavioral Scenarios:

5.1. A user story covers only one aspect of the goal or behavioral scenario. For example elaborating Step 4 "Create Release Vehicle" in Table 2 can be elaborated with User Stories as follows:

- As a (human playing the role of) Release Admin, I must be able to add a new Release Vehicle;
- As a Release Admin, I must be able to change Release Vehicles;
- As a Release Manager, I must be able to see a list of Release Vehicles;
- As a Release Manager, I should not be able to edit a list of Release Vehicles;
- As a Release Manager or Release Admin, I should be able to sort a list of Release Vehicles into the ascending or descending order.

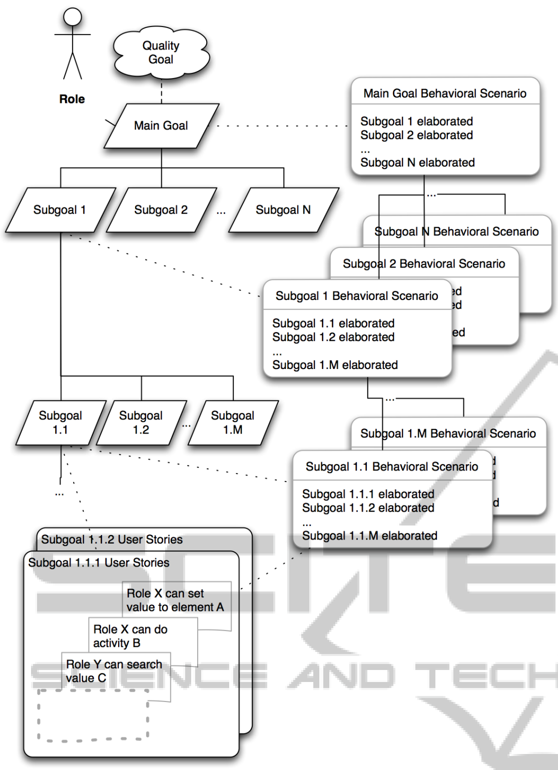


Figure 2: Concept of using models.

5.2. Mapping of terms from goals and behavioral scenarios to user stories is following:
   - Roles – a user story is always written from the perspective of a particular role.
   - Activity – activity defines how the corresponding goal is achieved. Performing an activity can be described by a number of implementable user stories.
   - Resource/ Knowledge Item – a resource of any kind required by actors playing Roles to achieve the goals set for the system. They may be knowledge items that actors playing Roles create or physical or virtual resources that are provided beforehand.
   - Additional comments on design – any kind of project-related information that is useful when describing and implementing user stories.
   - Step – provides a temporal view to be considered in user stories.
   - Condition – can be reflected in a user story as a Condition of Satisfaction or a separate user story (stories) that will cover the functionality of the condition.

6. Remember following remarks when creating models:

6.1. Completing all goal models at once is not required. For example, one branch of the goal model can be elaborated down to the lowest sensible level, while the rest of the branches are not. Other branches can be revisited and further elaborated later on.

6.2. Completing all behavioral scenarios at once is not required. Different independent branches of the overall goal model can be analyzed in parallel and at different paces.

6.3. Goal models can change according to the findings during the implementation of user stories.

6.4. Goal models and behavioral scenarios present an overall view of the system to be designed, describing what is required to be accomplished. User stories present specific design-related details for the system and link goals to concrete features of the system to be implemented.

6.5. An unlimited number of user stories can be created for each lowest behavioral scenario.

4 CONCLUSIONS AND FUTURE WORK

This was a nutshell description of our approach to RE by means of AOM and user stories. We claim that identifying, tracing and documenting is more simpler and lightweight with our method than full fledged RE practices. We have verified the simplicity of our approach by applying it in two real-life ASD projects. We acknowledge that we have not yet used any formal method or metric for measuring the efficiency of our approach. This is one of the major tasks for our future research work. We intend to investigate the relevant literature, e.g. (El Emam and Madhavji, 1995), and find an appropriate method for measuring the efficiency of our approach. However, so far we have a lot of overwhelmingly positive anecdotal evidence about successful application of our method. Participants in projects conducted by us have adopted the method and have successfully used it for eliciting and representing requirements, as well as for turning requirements into user stories for design and implementation.

The future work to be performed by us will include presenting our findings from the projects as detailed case studies. This should validate our findings with hard evidence. We have also been working on a prototypical RE and design environment that would accommodate our method. The prototype will be based on a wiki and will include an issue and project tracking system to add some flexibility and visibility to the RE and design processes.

REFERENCES

Alliance, A. (2013). Manifesto for agile software development. http://www.agilealliance.org/the-alliance/the-agile-manifesto/. [Online, accessed May 2013].

Ambler, S. W. (2002). *Agile Modeling*. John Wiley & Sons.

Beck, K. (2008). Tools for agility - a white paper. http://www.microsoft.com/en-us/download/details.aspx?id=4401. [Online, accessed May 2013].

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., and Perini, A. (2004). Tropos: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems*.

Cao, L. and Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *IEEE Softw.*, 25(1):60–67.

Cockburn, A. (2002). *Agile Software Development*. Addison-Wesley.

Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. The Addison-Wesley Signature Series. Addison-Wesley.

Dardenne, A., van Lamsweerde, A., and Fickas, S. (1993). Goal-directed requirements acquisition. In *Science of Computer Programming*, pages 3–50.

El Emam, K. and Koru, A. (2008). A replicated survey of it software project failures. *Software, IEEE*, 25(5):84–90.

El Emam, K. and Madhavji, N. (1995). Measuring the success of requirements engineering processes. In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on*, pages 204 – 211.

Group, S. (1995). Chaos report. https://cs.nmt.edu/~cs328/reading/Standish.pdf. [Online, accessed May 2013].

Haugset, B. and Stalhane, T. (2012). Automated acceptance testing as an agile requirements engineering practice. In *Proceedings of the 2012 45th Hawaii International Conference on System Sciences*, HICSS '12, pages 5289–5298, Washington, DC, USA. IEEE Computer Society.

Hull, E., Jackson, K., and Dick, J. (2011). *Requirements Engineering*. Springer.

Kazyrevich, A. (2010). Distributed agile development. http://codevanced.net/page/Talks-Distributed-Agile-Development.aspx. [Online, accessed May 2013].

Kotonya, G. and Sommerville, I. (1998). *Requirements engineering: processes and techniques*. Worldwide series in computer science. John Wiley & Sons.

lan Sommerville and Sawyer, P. (1997). *Requirements Engineering - A Good Practice Guide*. John Wiley & Sons.

Miller, T., Pedell, M., Sterling, L. S., and Lu, B. (2011). Engaging stakeholders with agent-oriented requirements modelling. *Agent-Oriented Software Engineering*, 6788(XI).

Miller, T., Pedell, S., Sterling, L., Vetere, F., and Howard, S. (2012). Understanding socially oriented roles and goals through motivational modelling. *J. Syst. Softw.*, 85(9):2160–2170.

Paetsch, F., Eberlein, A., and Maurer, F. (2003). Requirements engineering and agile software development. In *Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, WETICE '03, pages 308–, Washington, DC, USA. IEEE Computer Society.

Ramesh, B., Cao, L., and Baskerville, R. (2010). Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20(5):449–480.

Sillitti, A. and Succi, G. (2005). Requirements engineering for agile methods. In *Engineering and Managing Software Requirements*, pages 309–326. Springer Berlin Heidelberg.

Sommerville, I. (2010). *Software Engineering*. Addison-Wesley, Harlow, England, 9. edition.

Sterling, L. and Taveter, K. (2009). *The Art of Agent-Oriented Modeling*. MIT Press.

Taveter, K., Du, H., and Huhns, M. N. (2012). Engineering societal information systems by agent-oriented modeling. *J. Ambient Intell. Smart Environ.*, 4(3):227–252.

van Lamsweerde, A. (2001). Goal-oriented requirements engineering: a guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249 –262.

van Lamsweerde, A. (2009). *Requirements Engineering - From System Goals to UML Models to Software Specifications*. Wiley.

Vanhanen, J., Mantyla, M., and Itkonen, J. (2009). Lightweight elicitation and analysis of software product quality goals: A multiple industrial case study. In *Third International Workshop on Software Product Management (IWSPM)*, pages 42 –52.

Wooldridge, M. (2001). *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., New York, NY, USA.

Yu, E. S.-K. (1996). *Modelling strategic relationships for process reengineering*. PhD thesis, Toronto, Ont., Canada, Canada. UMI Order No. GAXNN-02887 (Canadian dissertation).