Evaluating A Novel Agile Requirements Engineering Method: A Case Study

Tanel Tenso, Alex Norta and Irina Vorontsova
Department of Informatics, Tallinn University of Technology, Tallinn, Estonia

Keywords: Visualization, Agile, Requirements Engineering, Modelling, Goals.

Abstract: The use of agile methods during software development is a standard practice and user stories are an established way of breaking complex system requirements into smaller subsets. However, user stories do not suffice for understanding the bigger picture of system goals. While methods exists that try to solve this problem, they lack visual tool support and are too heavy for smaller projects. This article fills the gap by evaluating a novel agile agent-oriented modelling (AAOM) method for requirements engineering. The AAOM-method comprises a visual approach to agile requirements engineering that links goal-model creation techniques taken from agent-oriented modelling and connects goals intuitively to user stories. A case study based evaluation explores the applicability of AAOM for requirements engineering in an agile software development process.

1 INTRODUCTION

Requirements engineering (RE) is an important software-development activity and traditionally considered one of the first phases in software development. RE is a process of formulating, documenting and managing the requirements for software and comprises requirements identification, analysis, documentation and validation (Hull et al., 2010). Since RE is the first software-development phase, late detected errors are very costly (Carlson and Matuzic, 2010) and produce incorrect software that does not satisfy customer needs.

Agile development is the most widely used method (Version One, 2016) for developing software systems that adhere to the agile manifesto (Beck et al., 2001). Agile development methods are time-boxed, iterative and incremental. Characteristic is also a frequent delivery of usable software and collaboration with customers. Additionally, the agile method supports self-organizing cross-functional teamwork. These factors play a role in the ability to quickly respond to changes (Cao and Ramesh, 2008). For agile affiliated RE, several variations exist that affect the use of RE, e.g., Scrum (Schwaber, 2004), XP (Beck, 2000), Lean (Poppendieck and Poppendieck, 2007), Kanban (Kniberg and Skarin, 2010). Common for these agile variations are of a lacking intuitive alignment (Cao and Ramesh, 2008) between engineered requirements and intuitive visual system development support.

To address this gap we define a novel requirements engineering method, namely the agile agent-oriented modelling (AAOM) method (Tenso and Taveter, 2013). AAOM emerges through multiple experiments in various projects varying from small scale projects, for example adding release management capability for a task management system, to large scale distributed projects like crisis simulation software development. AAOM is derived from agent-oriented modelling (AOM) (Sterling and Taveter, 2009), a holistic method for analysing and designing socio-technical systems consisting of humans and technical components. More concretely, AAOM focuses on a specific model type out of a larger model set that are part of AOM, namely goal models comprising functional goals, quality goals that are also knows as non-functional goals, and affiliated roles these goals affect.

To evaluate the applicability and usefulness of AAOM, we choose a case study based research methodology (Runeson et al., 2012; Yin, 2013). We use as a running case a project for developing a lost&found (l&f) mobile app. The business idea of the mobile app is to reunite lost objects of any type with their rightful owners. Instead of having to rely on lost&found offices at police stations, airports, cinemas and so on, the l&f-app is a simple and quick mobile solution to report findings by using smartphone capabilities: a phone camera allows to instantly take a picture of a found item while simultaneously the photo-shot location is traced and stored via smartphone GPS. The app is also beneficial for people who lose something: the app announces the loss and receives notifications when an item with a similar description and attributes is found and entered into the database. The l&f-app either establishes the association between object and owner because of provided descriptions, or the object is tagged with a visual identification code a mobile phone can read. The l&f-app also offers a so-called giveaway section for found items that nobody claims.

The remainder of the paper is structured as follows. Earlier studies and theory of AAOM are in Section 2. Research questions, case setup, data collection, analysis and validity procedures are described in Section 3. In Section 4, we present results and findings of the analysis of collected data. Finally, Section 5 concludes findings and gives open issues for future work.

2 RELATED WORK

To provide a basis for the evaluation, in Section 2.1, we give more details about the application of AAOM. In Section 2.2, we discuss similar methods to AAOM for comparison.

2.1 AAOM-Method Explanation

Face-to-face communication in agile over written specifications facilitates embracing change and applying iterative cycles for RE. Figure 1 depicts (Tenso and Taveter, 2013) how AAOM activities fit into the agile development lifecycle. After an initial discovery phase, the result is a set of further elaborated goal models and affiliated user stories for which a preliminary backlog is established. Next, in iterations, the goal models and their user stories evolve by changing, eliminating, updating the latter. Once the main development phase of a project ends, the system-maintenance phase commences where the goal models and user stories are the foundation for exploring how to add, modify, or remove features.

The AAOM method links goal models from AOM (Sterling and Taveter, 2009) to user stories in accordance with Figure 2. In the depicted model, goals are shaped as parallelograms and quality goals shaped as clouds represent functional- and non-functional system requirements respectively. Goal models also contain roles as sticky men with relationships between roles and goals/quality goals. User stories are attached in Figure 2 at the leave-level of sub-goals for

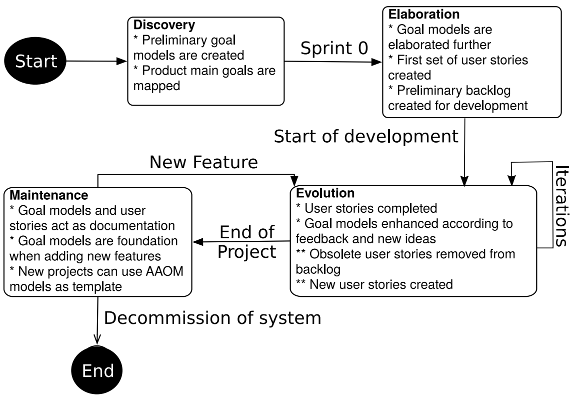
Start
Discovery
* Preliminary goal models are created
* Product main goals are mapped
Sprint 0
Elaboration
* Goal models are elaborated further
* First set of user stories created
* Preliminary backlog created for development
Start of development
New Feature
Iterations
Maintenance
* Goal models and user stories act as documentation
* Goal models are foundation when adding new features
* New projects can use AAOM models as template
End of Project
Evolution
* User stories completed
* Goal models enhanced according to feedback and new ideas
** Obsolete user stories removed from backlog
** New user stories created
Decommission of system
End

Figure 1: AAOM activities in agile development lifecycle.

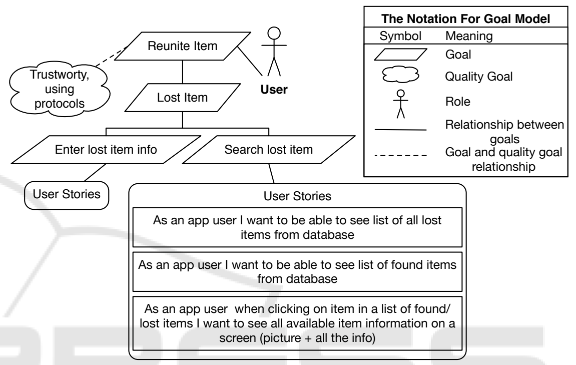
Reunite Item
Trustworty, using protocols
User
Lost Item
Enter lost item info
Search lost item
User Stories
User Stories
As an app user I want to be able to see list of all lost items from database
As an app user I want to be able to see list of found items from database
As an app user when clicking on item in a list of found/ lost items I want to see all available item information on a screen (picture + all the info)
The Notation For Goal Model
Symbol
Meaning
Goal
Quality Goal
Role
Relationship between goals
Goal and quality goal relationship

Figure 2: Example of a goal model with user stories attached.

establishing and tracing a connection to a system's top goal.

User stories are simple artefacts for agile software development and requirements documentation (Cohn, 2004). A user story is a written sentence or two that describe functionality from a system's user point of view. There are several formats and concepts in which Cohn's (Cohn, 2004) definition is as follows:

As a <*role/type of user*>, I want <*goal/desire*> so that <*benefit/reason*> .

Examples:

As a *user*, I want *to reserve a hotel room*.

As a *frequent flyer*, I want *to rebook a past trip*, so that *I save time booking trips I take often*.

A user story must be small enough for implementation within one development iteration (Cohn, 2004). This limits implementation work per user story, providing a fast feedback and verification of requirements for system development.

Similar to AAOM, other methods exist, that provide structure to agile requirements by organizing user stories. These methods are briefly discussed in the next section.

2.2 Earlier Studies

Similar to AAOM, there are methods for organizing user stories into structures to mitigate visibility problems, for example Cohn (Cohn, 2004) suggests Epics, that are bigger user stories grouping smaller ones. Epics covering different levels of abstraction are also used in Scaled Agile Framework(SAFe) (Leffingwell, 2016). Large-Scale Scrum (LeSS) (Larman and Vodde, 2008) concentrates on splitting requirements into small Product Backlog Items, usually user stories. Disciplined Agile Delivery (DAD) (Ambler and Lines, 2012) sums up many agile practices and introduces the term Portfolio management that is requirements management in a hierarchical list of work items. Scrum of Scrums (Sutherland et al., 2007) includes team level planning and requirements tracking between teams. The lean approach to agile requirements (Leffingwell, 2010; Leffingwell and Aalto, 2009) divides requirements according to details onto team-, program- and portfolio level. Despite having several positive impacts and influencing AAOM theory aspects, these approaches are not truly visual and, according to our experience not graspable without special training by non IT stakeholders. Furthermore, these methods are meant for enterprise scale usage and are too heavyweight for smaller projects where it is important to establish a conversation with clients and align everybody to the same set of goals.

On the other hand, goal modelling techniques exist exactly for depicting system goals in a visual way. Goal-based requirements engineering is well established, for example Hull et al. (Hull et al., 2010) suggest representing use scenarios as a sequence of goals. Lamsweerde approaches goal modelling from a formal point of view, providing a mathematical proof and meta model for goal based requirements engineering. His effors culminate in (Van Lamsweerde et al., 2009), covering high-level system analysis with goal techniques. One prominent method that includes goal modelling is i* (Yu, 2009) that provides both tooling and principles for the problem domain. Finally, an example of more social goal-modelling techniques is described in agent-oriented modelling (AOM) (Sterling and Taveter, 2009) theory that is the basis for AAOM. However, the aforementioned approaches are too heavyweight to be included into short agile development feedback cycles and are meant for model driven development. While AOM provides the most lightweight goal modelling technique for relating roles, functional- and non-functional requirements, its other models focus on the agent paradigm and are thus too specific for wider system-design use. Connecting agile user stories with goal models is a novel approach introduced by AAOM.

3 CASE STUDY DESIGN

We choose a case study research method to conduct the evaluation of the AAOM-method. First we define research questions in Section 3.1 for guiding the AAOM evaluation, followed by a justification of case- and subject selection in Section 3.2. Data sources of evidence are discussed in Section 3.3, followed by an analysis procedure in Section 3.4. A validity discussion finalizes the case study design in Section 3.5.

3.1 Research Questions

Based on previous experiments with the AAOM method and feedback gathered during them, the following main research question is devised:
How does AAOM help to improve software engineering requirements engineering activities?
This question can be refined in various sub-questions:
RQ1: What are the benefits of using AAOM from a user perspective?
RQ2: What effect has project setup and tooling on AAOM usage?
RQ3: What aspects of the AAOM-method usage need further refinement?
Aforementioned questions establish the basis for selecting appropriate case, data sources and analysis methods.

3.2 Case Selection

The l&f-app development project follows an agile software development cycle, specific scrum techniques such as planning with user stories, backlog management and iterative development are used. Three development iterations take place in which work is visualized on a task board and meetings with clients take place at the end of iterations.

For the l&f-project research, we set up as a single case study with holistic design (Runeson et al., 2012). The unit of analysis and the case is an AAOM RE method application for iterative requirements gathering. Four people participate in this project carrying the labels role and experience. Role helps us to evaluate the AAOM usage by different team members, experience provides reliability for gathered information. Three participants fill the role of client who order the l&f-app development, whereas one of them has a high level of experience as an ICT expert while the other two are from different domains. One person acts as an analyst and developer, being a beginner in the first role and highly experienced in the second. A high level of experience means a person acts at least for two years in a specific role.

The research team consists of three researchers working in co-operation, providing peer-review to each other. The procedures to observe a case include taking part in all meetings between case subjects that are modelling, demo and retrospective sessions while video recoding them all. We act as silent participants taking notes of the AAOM usage throughout different meetings. Based on the research questions and meeting notes, we devise interview questions and sessions for gathering qualitative data. The analysis of the gathered data by researchers provides answers to the research questions.

3.3 Data Sources

Interviews are the most valuable data source for the running case study. Based on recommendations from (Runeson et al., 2012), interview planning commences with selecting interviewees. There are four people and three roles in the running case as mentioned in Section 3.2 and all participants are interviewed. Since there is one person in both roles of analyst and developer, different questions are posed to her addressing both roles. Next follows the planning of interview sets. Ideally, interviews take place multiple times, for example after elaborating a first branch of requirements, after every development iteration and at the end of the project. As a limitation, for this l&f-app project, we conduct only one set of interviews after completing the planning session and three development iterations.

For conducting interviews, we choose a semi-structured format (Runeson et al., 2012). The research questions stated in Section 3.1 and the meeting notes guide the preparation of interview questions. Different sets of questions target each respective role without offering predefined answers. Thus, interviewees can not answer, e.g., ”yes” or ”no”, instead they must express their own opinions. Questionnaires for client, analyst and developer roles are depicted in a longer version of this article[1].

The interviews adhere to a time-glass model (Runeson et al., 2012) so that an interview begins with broad questions first and continues with more specific questions. At the end of an interview, again broad questions are presented. Thus, four interviews include three with clients and one combined for the developer/analyst. The interview structure is similar in all cases and the interviewees are informed about the interview structure during the process.

Each interview session lasts roughly one and a half hours and starts with an introduction, followed by role specific questions. The interviews are audio recorded into MP4 files for subsequent post-interview activities and analysis. In case an interviewee responds to a question briefly, we ask additional questions on the same topic to gather more insight. At the end of an interview, a participant learns that every interview is transcribed and sent for verifying the captured ideas are correct.

We also gather work artefacts, such as goal models, user stories and source code. Since a dedicated development toolkit for AAOM does not exist yet, we employ provisionally a set of freely available tools to host our case. The first tool, Draw.io[2], is an online diagramming tool to draw goal models for the l&f-app. Another used tool is Trello[3], a collaboration tool to organize project tasks on boards. Trello visualizes tasks in the form of user stories similar to post-it notes in status columns to observe the progress during software development, e.g., to do, pending, in progress, completed, and so on.

The generated goal models and user stories are relevant for investigating the evolution during the project while the history of changes is recorded. Project source code for the l&f-app in a version control system allows to observe lines of code (LOC), changes in LOC, time between LOC changes and links to user stories. After evaluating these quantitative data sources, the main finding is that there is no existing body of knowledge for analysing them as needed for the AAOM method usage validation. Thus, we use these anecdotal data sources only to subjectively evaluate some statements received from interviewees.

As our main data source is interviews, we focus in the next section on interview analysis. The latter yields answers to the research questions.

3.4 Analysis Procedure

Interview analysing requires first transcribing and then coding (Saldaña, 2015) the results and for both we use the tool NVivo[4]. We transcribe the interviews and for corrections and clarifications, the interviewees review the transcripts.

To code the interviews, we determine first a list of so-called *a priori codes* (Saldaña, 2015) that are

[1]http://digi.lib.ttu.ee/i/?4064

[2]https://www.draw.io

[3]https://trello.com

[4]http://www.qsrinternational.com/

deduced from the research questions. Upon labelling interviews with predefined labels, we detect sentences and blocks of text that do not fit into existing codes. The contents of these exceptions lead to the creation of new so-called called *grounded codes* (Saldaña, 2015).

To summarize coding, first themes are introduced (Saldaña, 2015) for grouping codes. This axial coding (Runeson et al., 2012) identifies connections between the themes and codes. Themes are also divided into two sets, either based on relations to research questions (Section 3.1), or for not having any relation.

Additionally, we perform theoretical coding for devising a set of attributes to codes for evaluating the latter during analysis. The chosen attributes attached to codes are *polarity* and *type*. Polarities show the opinion of the interviewee and has three possible values, namely positive, neutral or negative. Types show if a code and thus, the interviewee references an existing situation, or a need for change/addition. Type has two values, namely statement or suggestion.

Polarity- and type values provide combinations that describe whether a code proves, contradicts, or has no relation to research. The first combination of six, *negative suggestion*, indicates a need to change an existing part of case under investigation. Second, *positive suggestion* is an additional idea, or improvement put forward by the interviewee. Next, *neutral suggestion* is not related to the AAOM method evaluation, *negative statement* denotes a flaw in the AAOM-method while the interviewee has no improvement suggestion. The final two combinations are *positive statement* for indicating a participant's satisfaction with AAOM and *neutral statement* for representing an opinion, or a statement of affairs not related to the research question.

We use a simple formula to evaluate which codes have more value for analysis. Three components play a role in determining code validity, namely first *references* that show how many times a code is mentioned in interviews. The second element is *sources* denoting how many different interviewees mention one code. Finally, *role experience* expresses an interviewee's experience in a role. Each component has numeric values and a higher value increasing code validity. The formula to sort codes based on mentioned components is as follows:

$$\boxed{codevalue = (references * sources) + experience}$$

3.5 Validity Procedure

To asses the validity of our research, we use criteria proposed by A. K. Shenton (Shenton, 2004), namely credibility, transferability, dependability and confirmability. To increase credibility, we employ the following strategies:

- We use a well established body of knowledge of conducting case studies, mainly Runeson et al (Runeson et al., 2012). To set up and receive useful data from interviews, we use guidelines by Robson (Robson, 2002) and for best-practices to analyse gathered data, we consider Saldana (Saldaña, 2015).
- We conduct interviews with all participants in the project covering all different roles. This is considered as a form of triangulation since we capture viewpoints of all informants (Van Maanen, 1979). Unfortunately, we are not able to triangulate via other types of data sources, since there is a lack of methods for analysing goal models.
- Before starting the l&f-app development, researchers study AAOM as the unit of analysis and also the context where it is applied, the field of finding and losing assets.
- To help ensure honesty in subjects, we inform them that the data use is anonymous and that their voice is recorded. All interviewees agree with recording. The interviewees review the transcribed documents to assure a valid transferral of ideas.
- Three researchers participate in the case study and provide constant peer reviewing to each other. With that setup and constant debriefing among each other, the researchers' vision is wider than working alone.
- The researchers participating in the case study have relevant related backgrounds. Thus, credibility is assured by the extensive research experience.

The second criteria, transferability, can not be demonstrated since the case covers only a specific project and a specific set of individuals (Shenton, 2004). Still, the reporting context of this case study is helpful for other researchers to compare their results with ours (Shenton, 2004). To test transferability, we conduct another case study with a similar setup to verify whether the same results reoccur. One more factor assuring transferability is the fact that agile teams are limited by definition to a size of 3 to 9 (Cockburn, 2006).

To address dependability, Shenton et al. (Shenton, 2004) recommend a case study report that includes sections devoted to:

- the research design and its implementation
- the operational detail of data gathering

- reflective appraisal of the project

In this article, the research design is described in Section 3 and data gathering details are depicted in Section 3.3. We do not evaluate the effectiveness of processes undertaken in the current case study. This will be a evaluated by a future case study.

According to (Jensen, 2008), confirmability is an accurate means through which to verify the two basic goals of qualitative research:

- to understand a phenomenon from the perspective of the research participants and
- to understand the meanings people give to their experiences

This research contains a threat to researcher bias because one researcher is the inventor of AAOM, a method under investigation. We are aware of this threat and avoid it with the same means as for credibility - by providing detailed descriptions. According to (Shenton, 2004), in a qualitative study, researchers biases are inevitable. We also acknowledge the limitation of one person acting in two roles possibly removing one friction point between analyst and developers. This could have been mitigated by assigning another person to that role for which we lack a sufficient project budget.

4 RESULTS

The main goal of the analysis is to understand whether theories about the AAOM method are valid by finding answers to questions specified in Section 3.1. We also take into account any non-expected data found during the interviews. All themes and codes discussed further on, are presented in longer version of this article[5].

4.1 Benefits of using AAOM From a User Perspective (RQ1)

In order to find answers to RQ1, we use five themes that cover direct benefits the participants state about how AAOM improves communication by collaborative modelling by including the participants. Additionally, a method comparison along with visual representation provides insights to AAOM.

To evaluate results, we refer to a formula devised in Section 3.4. The first theme *Benefits* shows all codes are positive statements for using AAOM. All the codes in this theme are positive statements and adhere to the case-study research question about what the benefits are of using AAOM. The highest ranked codes are a secure feeling for the project direction, mutual communication and discovering new angles in requirements for the project. Four codes have a lower evaluation score and thus, are more unreliable to draw confident conclusions from.

Next, theme *Collaborative Modelling* relates to the AAOM-theory of improving communication between clients and the development team by working together on requirement elicitation. The highest rated codes are all positive suggestions and thus, confirming the expectations set by AAOM. Having everyone "on the same page", improves understandability and pinpointing problems represents a benefit for collaborative modelling. There is one negative suggestion about composing goal models should be more structured.

The *Method Comparison* theme gathers the participants' experience with similar methods compared to AAOM. Unfortunately, the experience with similar methods is low among participants in the running case study. Consequently, a comparison with other methods is not sufficient and does not provide enough results.

The final two themes have less codes than *Benefits* and *Collaborative Modelling* while the codes have a high formula value that shows same opinions from all participants. The *Participation* theme gathers objective opinions about how the method includes everyone in the project and it shows the ICT-expertise is an advantage. The theme *Visual Representation* consists of only one strongly referenced code denoting that the goal-model representation is a definite benefit.

4.2 Effects of the Projects Setup and Tooling on AAOM (RQ2)

Three themes in Section 3.1 cover RQ2 starting with explaining the setup of elaboration sessions and finding the effects for the AAOM-method application. The next theme is related to temporal measures as well, explaining time usage to manage AAOM models. The final theme covers effects of software based tooling usage on AAOM application.

Which practices are favoured and which need improvement, are ranked again by formula results (Section 3.4). The *Elaboration Sessions* theme covers the AAOM-method's application-session content, -duration and -suitability. The highest ranked code suggests the selected session length that is 1.5 hours, is selected correctly even though one low ranked contracting code is gathered to have shorter sessions. The remaining codes are ranked relatively low with inconclusive findings.

[5]http://digi.lib.ttu.ee/i/?4064

The theme *Modelling Time Usage* answers to questions related to spent time on modelling activities carried out using the AAOM method. Positive statements are gathered for system requirements fast capture, fast development based on models and overall effective time usage. One neutral statement is added that moderate time is spent until an idea is formed as a user story. The unit of measurement is the participants' subjective feelings about the time spent on method activities.

In order to answer aspects of tool importance in using AAOM, we find a theme *Tools Usage*. There are several different codes revealing several viewpoints. The two highest ranked codes show that using freely available tools is provisionally satisfactory while having an integrated suit would reduce the amount of work needed for completing tasks. Available commercial tools are considered better while they are prohibitively expensive, especially in smaller projects. We conclude interest exists to use new tools that are better tailored than the chosen free tools in the running case.

4.3 Further AAOM-refinement Meeds (RQ3)

For further improvements we collect codes under only one theme called *Method Clarification* that addresses what practices of AAOM are clear and which need explanation, or redefining. On the positive side, we find that the sequence of activities for goal models composing is clear. In top ranked codes we also find positive statements being clear about concepts of quality goals, user stories and roles.

On the negative side, we gather contradicting information if creating user stories for the lowest level goals is clear. Both codes are highly ranked while statements about the process being unclear are ranked slightly higher. Studying this contradiction is future research work in addition to the exact usage of quality goals in relation to user stories that is currently unclear. As a counterpart for top ranked positive statements, there are few contradicting negative statements about same aspects. Analysing these contradictions deeper, we find that a lack of experience causes these results.

4.4 Emerged Results

The emerged codes result from grounded coding that is explained in Section 3.4. The participants express their feelings in the theme *Drawbacks* about the project setup that is not directly related to AAOM modelling while it still affect its usage. The most mentioned code states that experienced participants are needed to fully benefit from the AAOM-method. Thus, AAOM is not intuitively fully understandable to all participants in requirements engineering. To aid understandability, a solution could be a better user guide for the AAOM method, as suggested by interviewees in the second top mentioned code. Furthermore, the analyst is identified as carrying a vital role in AAOM usage by having the biggest responsibility in modelling the activities. As a final statement worth noting a doubt exists pertaining to the method suitability in smaller projects due to a possible method overhead.

The *Expectations* theme shows the participants' positive impression. Working results and extensible implementations are expected, goal models and user stories are expected to be updated. A negative point is that more participants are needed to accomplish the goals pertaining to the project setup and not the implementation phase of AAOM.

The results captured under the theme *Modelling Suitability* support the theory about the method helping to focus on objectives elicitation and organizing thoughts to express a client's feelings. Statements gathered under the theme *New Ideas* give novel suggestions stemming from the study participants such as assigning financial values to the goal, to use goal models as a system documentation, and so on.

5 CONCLUSION

In this article, we evaluate the AAOM-method as a novel method for requirements engineering in a small-scale project that employs agile software development for developing a mobile app from scratch. A case-study based research approach is instrumental for an evaluation. Interviews with project participants provide the most relevant input for the AAOM-method's evaluation. Interviews are coded and analyzed to answers research questions, also new knowledge outside the scope of research questions we generate.

The benefits of using the AAOM-method are a secure feeling for project direction, mutual communication and discovering new angles in requirements for the project. The visual representation provided by goal models is intuitively comprehensible. Furthermore, the AAOM-method enforces communication via collaborative modeling, improving understandability, pinpointing problems and involving participants. Time spent on AAOM activities for gathering requirements is found to be adequate and the overhead during that process is marginally low. For projects on a small budget, free tools with manual integration suffice while there is a desire for AAOM-tailored integrated tool support. Unclear procedures, that need better guidance or redefining, pertain to the way of finding the lowest level of goal models with the corresponding deducing of user stories, and quality-goal use. One of the discovered factors outside the scope of research questions is that before using AAOM effectively, longer experience in ICT is needed or additional guidance should be available. Finally, the analyst role has been identified as too heavy in applying the AAOM method.

The first limitation is that the collected information is mostly based on the client-side opinions because three participants out of four fill the client role. Only two of the four participants have experience participating in software development processes and consequently, feedback also stems from inexperienced participants. Another limitation is a lack of interview feedback with respect to other methods for agile and requirements/engineering methods. This can reveal how well the AAOM-method is combinable with other agile methods. A final limitation is related to research bias towards supporting factors of AAOM usage since one researcher is an inventor of AAOM.

Further studies and investigation must apply the AAOM-method in diverse domains for demonstrating its universal applicability. Based on the feedback from participants of the running case, an improvement of the AAOM-method must include better explanation of the role of quality goals. Since the analyst plays an important role during applying the AAOM-method, we need a deeper understanding of this essential role.

REFERENCES

Ambler, S. W. and Lines, M. (2012). *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. IBM Press.

Beck, K. (2000). *Extreme programming explained: embrace change*. Addison-Wesley.

Beck, K. et al. (2001). The agile manifesto.

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

Carlson, D. and Matuzic, P. (2010). Practical Agile Requirements Engineering. Technical report.

Cockburn, A. (2006). *Agile software development: the cooperative game*. Pearson Education.

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

Hull, E. et al. (2010). *Requirements engineering*. Springer.

Jensen, D. (2008). *Confirmability*, page 113. Sage.

Kniberg, H. and Skarin, M. (2010). *Kanban and Scrum-making the most of both*. Lulu.com.

Larman, C. and Vodde, B. (2008). *Scaling lean & agile development: thinking and organizational tools for large-scale Scrum*. Pearson Education.

Leffingwell, D. (2010). *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley.

Leffingwell, D. (2016). Scaled agile framework. *http://scaledagileframework.com*.

Leffingwell, D. and Aalto, J. (2009). A lean and scalable requirements information model for the agile enterprise. *https://scalingsoftwareagility.wordpress.com/*.

Poppendieck, M. and Poppendieck, T. (2007). *Implementing lean software development: from concept to cash*. Pearson Education.

Robson, C. (2002). *Real word research*. Blackwell.

Runeson, P., Host, M., Rainer, A., and Regnell, B. (2012). *Case study research in software engineering: Guidelines and examples*. Wiley.

Saldaña, J. (2015). *The coding manual for qualitative researchers*. Sage.

Schwaber, K. (2004). *Agile project management with Scrum*. Microsoft Press.

Shenton, A. K. (2004). Strategies for ensuring trustworthiness in qualitative research projects. *Education for Information*, 22(2):63–75.

Sterling, L. and Taveter, K. (2009). *The art of agent-oriented modeling*. MIT Press.

Sutherland, J. et al. (2007). Distributed scrum: Agile project management with outsourced development teams. In *System Sciences*. IEEE.

Tenso, T. and Taveter, K. (2013). Requirements engineering with agent-oriented models. In *Evaluation of Novel Approaches to Software Engineering*, pages 254–259.

Van Lamsweerde, A. et al. (2009). *Requirements engineering: from system goals to UML models to software specifications*. Wiley.

Van Maanen, J. (1979). The fact of fiction in organizational ethnography. *Administrative Science Quarterly*, pages 539–550.

Version One, I. (2016). 9th annual state of agile survey. *http://info.versionone.com/state-of-agile-development-survey-ninth.html*.

Yin, R. K. (2013). *Case study research: Design and methods*. Sage.

Yu, E. S. (2009). Social modeling and i*. In *Conceptual Modeling: Foundations and Applications*, pages 99–121. Springer.