Evaluating Novel Agile Requirements Engineering Method: A Case Study

Tanel Tenso[1], Alex Norta[1], Irina Vorontsova[1]
[1]*Department of Informatics, Tallinn University of Technology, Tallinn, Estonia*
tanel.tenso@ttu.ee, alex.norta.phd@ieee.org, ivorontsova@gmail.com

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 (I. Sommerville, 1997),(Hull et al., 2005) and comprises requirements identification, analysis, documentation and validation (I. Sommerville, 1997). Since RE is the first software-development phase, late detected errors are very costly (Leffingwell and Widrig, 2000; Carlson and Matuzic, 2010) and produce incorrect software that does not satisfy customer needs.

RE may unfold in variations depending on the software-development method it is coupled with. In the case of waterfall-method affiliation (Neill and Laplante, 2003), RE is presented as the first phase of the development process. Agile development is the most widely used method (Version One, 2015) for developing software systems. Agile software development is a group of software development methods that adhere to the agile manifesto (Beck et al., 2001). For example, (Cao and Ramesh, 2008) describe agile development methods as 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.

In modern agile software-development projects (Cao and Ramesh, 2008), RE continues through the lifetime of a system. For agile affiliated RE, again several variations exist that also 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 problems (Cao and Ramesh, 2008) of a lacking intuitive alignment 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 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 that comprise functional goals, quality goals that are also knows as non-functional goals, and affiliated roles that these goals affect. The AAOM method in its current state is described in Section 2.

To evaluate the applicability and usefulness of AAOM, we choose a case study based research methodology (Runeson et al., 2012; Yin, 2013; Stake and Savolainen, 1995; Runeson and Höst, 2009). 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. As Figure 1 depicts (Tenso and Taveter, 2013), AAOM commences with a discovery of preliminary goal models for mapping system-product goals onto them. After an initial discovery phase called Sprint 0, the result is a set of further elaborated goal models and affiliated user stories for which a preliminary backlog is established. Next, in interactive feedback provisions, the goal models and their user stories evolve by changing, eliminating, updating the latter.

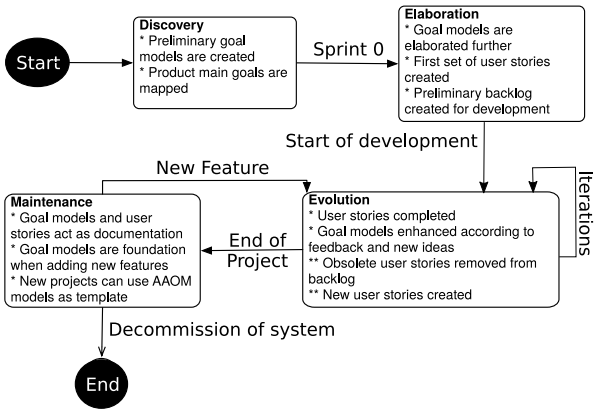

Figure 1: A lifecycle for AAOM application.

Once the main development phase of a project ends in Figure 1, the system-maintenance phase commences where the goal models and user stories are the foundation for exploring how to add, modify, or remove features.

We choose AAOM because the models are intuitively understandable for stakeholders, also for non-technical practitioners (Sterling and Taveter, 2009; Miller et al., 2011). As the sequel explains, AAOM links goal models 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 establishing and tracing a connection to a system's top goal.

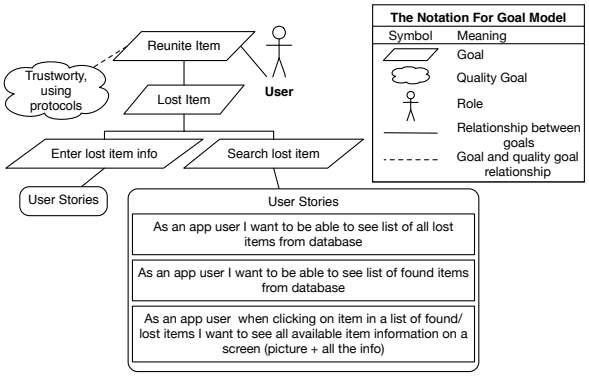

Figure 2: Example of a goal model with user stories attached

User stories are simple artefacts for agile software development and requirements documentation (Cohn,

2004; Paetsch et al., 2003). 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*> .

The last part "so that <*benefit*>" can be omitted in some cases if the goal describes the benefit/reason well enough.

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 while larger user stories must comprise smaller user stories (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, 2013). 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 term Portfolio management which 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 well established, for example Hull et al. (Hull et al., 2005) suggest representing use scenarios as a sequence of goals. (Dardenne et al., 1993) approach goal modelling from a formal point of view, providing a mathematical proof and meta model for goal based requirements engineering. More recent work by (Van Lamsweerde, 2001) elaborates and refines this mathematical model. Lamsweerde also includes goal modelling in his requirements engineering book (Van Lamsweerde et al., 2009), covering high-level system analysis with that technique. One prominent method in including goal modelling is i* (Eric, 2009) that provides both tooling and principles for the dissecting 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 as covered in this article. 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 AAOM helps 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: Which 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 in accordance with the AAOM method. While the presence of just one developer yields simple development processes, 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 with a scrum 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 thus the case, is an AAOM RE method application for iterative requirements gathering. Four people participate in this project whom we label with two 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, one person acts as an analyst and developer. One of the three clients has a high level of experience as an ICT expert while the other two customers are from different domains. Likewise, the developer is highly experienced and the analyst is not. A high level of experience means a person acts at least two years in a specific role.

The research team setup is simple, consisting of three researchers working in co-operation, providing peer-review to each other. 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 making 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 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 (Robson, 2002; Runeson et al., 2012). While planning the interview questions, their order is not of importance and changeable during the interview, depending on a discussion flow and interviewee answers to preceding questions. Semi-structured interviews can also provide additional insight beyond interview questions.

The research questions stated in Section 3.1 and researchers meeting notes guide the preparation of interview questions. Different sets of questions target each respective role (client, analyst, developer) without offering predefined answers. Thus, interviewees can not answer, e.g., ”yes” or ”no”, instead they must express their own opinions. Questionnaires for client role are represented in Table 1, for analyst in Table 2 and developer in Table 3

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. All together, in the current case study, 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

Table 1: Client questionnaire

Question
Did the goal models prove to be useful in subsequent conversation, over changes?
Was the implementation of the system satisfactory?
Did something specific about the implementation catch your attention?
Did you see that quality goals affect the implementation outcome?
Any ideas or questions about models, user stories and implementation alignment?
Was progress of implementation traceable for you?
Has the tooling satisfied your needs so far?
Did your idea about the project outcome change after implementation?
Any other ideas or remarks about the modelling method?

Table 2: Analyst questionnaire

Question
Did negotiating changes with clients work better with goal models, assuming there were changes?
How much did goal models change?
How much time did it take to reflect changes back to goal models?
If and how did quality goals affect user stories?
How much did existing user stories change?
Did developers understand the user stories?
How much did developers need extra clarification about user stories?
Has the tooling satisfied your needs so far?
Did your idea about the project outcome change after the implementation?
Any other ideas or remarks about the modelling method?

Table 3: Developer questionnaire

Question
When implementing user stories, did you receive many questions that couldn't be answered with goal models?
Were the user stories small enough for you to implement in a day or two?
When implementing user stories, did you have the question: "why is this functionality needed"?
Would you have liked your tasks to be presented in other ways?
Did your idea about the project outcome change after implementation?
Any other ideas or remarks about the modelling method?

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[1], is an online diagramming tool to draw goal models for the l&f-app. Another used tool is Trello[2], 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. Finally, BitBucket[3] is a free source code hosting service for Git and at the same time a simple wiki for storing documentation.

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 goal models. 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 of evaluating AAOM.

3.4 Analysis procedure

To analyse the interview requires first transcribing and then coding (Gibbs and Taylor, 2005; Saldaña, 2009) the results. To code the interviews, we determine first a set of labels or codes based on research interests. Codes are assigned to phrases, or sentences from the interviews in a second step. Additionally, sentences not fitting under predefined codes are added to code lists as grounded codes. To summarize coding, themes are introduced (Runeson et al., 2012) for grouping codes. Each theme corresponds to a research question from Section 3.1. In the sequel, a formula is devised to compare the validity of codes against each other for analysing and evaluating aspects of AAOM.

We transcribe the interview text files manually before analysing them. To allow for a final possibility

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

[2] https://trello.com

[3] https://bitbucket.org

for corrections and clarifications, the interviewees review the transcripts. As a next step, we use a coding technique to prepare transcripts of the interviews for the analysis phase. According to (Gibbs and Taylor, 2005; Saldaña, 2009), coding is defined as

> "the process of combing the data for themes, ideas and categories and then marking similar passages of text with a code label so that they can easily be retrieved at a later stage for further comparison and analysis. ... Coding the data makes it easier to search the data, to make comparisons and to identify any patterns that require further investigation."

Shortly, codes are meaningful keywords, or labels organized by themes, or categories. Each code represents a phrase, or few phrases from interviews. For coding, we use the tool NVivo[4] that is a qualitative data analysis software.

At the start, a list of so-called *a priori codes* (Gibbs and Taylor, 2005; Saldaña, 2009) are deduced from the research questions. Upon labelling interviews with predefined labels, we detect sentences and blocks of text that do not fit under existing codes. The contents of these exceptions lead to the creation of new codes called *grounded codes* (Gibbs and Taylor, 2005; Saldaña, 2009). The latter are a source of unexpected-, or novel findings.

The second cycle commences (Saldaña, 2009) with so-called axial coding where connections between the categories and codes are identified. 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*. Polarity denotes an emotionality of a code, and the type show if a code is a recommendation, or a remark from the interviewee.

Polarity has three possible values showing the opinion of the interviewee, namely positive, neutral or negative. Type has two values, namely statement or suggestion. The former means that an interviewee references an existing situation and the latter references a need for change, or addition.

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 participants' satisfaction with AAOM and *neutral statement* for representing an opinion, or a statement of affairs not related to the research question.

The final step in preparing the interviews for analysis is post-coding to establish theme groups (Saldaña, 2009). Themes are also divided into two sets, either based on relations to research questions (Section 3.1), or for not having any relation.

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:

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

To justify our case selection, data collection procedures and analysis procedures, we discuss validity procedures next.

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 Gibbs and Saldana (Gibbs and Taylor, 2005; Saldaña, 2009).
- 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 context where it is applied, the field of finding and losing assets.

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

- 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 then 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, 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 and Highsmith, 2001).

To address dependability, Shenton et al. (Shenton, 2004) recommend a case study report 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 depicted in Section 3.3. Evaluating the effectiveness of processes undertaken in the current case study, is not carried out. 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.

The next section focuses on finding answers to the research questions posed in the beginning of current section.

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 presented in subsequent tables have the same notation. The theme/code column contains themes spanning over all columns and codes. Polarity- and type-attribute values we abbreviate, i.e., polarity value *P* denotes positive, *N* stands for neutral and *E* stands for negative. The type column values are respectively *S* for statement and *U* for suggestion. The value column gives the analysis formula result. The reminder of the current section displays the results from the code analysis.

4.1 Benefits of using AAOM from a user perspective (RQ1)

In order to find answers to RQ1, we use the themes in Table 4. With these five themes, we cover direct benefits that participants state about how AAOM improves communication by collaborative modelling and the inclusion of participants. Also a method comparison along with visual representation provides insights to AAOM.

To evaluate results, we refer to a formula devised in Section 3.4. A detailed list of codes for RQ1 related themes is in Table 5. 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 are the benefits of using AAOM. The highest ranked codes are a secure feeling for a 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, 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 amongst participants in the running case study. As the result, 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 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.

The next section discusses how the project setup and tooling affect usage of AAOM.

Table 4: AAOM usage benefits themes

Theme name	**Theme**
Benefits	What are the benefits of using AAOM?
Collaborative modelling	How to improve communication between participants?
Method Comparison	How does AAOM competitively compare to other methods?
Participation	Does AAOM include participants?
Visual Representation	Is a visual approach suitable for RE?

4.2 Effects of the projects setup and tooling on AAOM (RQ2)

Table 6 introduces themes related to RQ2 raised in Section 3.1. Three themes cover this research question, starting by explaining the setup of elaboration sessions and finding the effects for 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.

To evaluate which practices are favoured and which need improvement, we use again formula results (Section 3.4), details are presented in Table 7. The *Elaboration Sessions* theme covers the AAOM-method's application-session content, -duration and -suitability. The highest ranked code suggests that the selected session length, which 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.

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

Table 5: AAOM usage benefits codes

Theme/Code	**Polarity**	**Type**	**Value**
Benefits			
Secure feeling for project direction	P	S	10
Mutual communication	P	S	7
Discover new angles	P	S	4
Intuitively understandable	P	S	2
Easily modifiable	P	S	2
Constructive modelling	P	S	1
Estimate work ahead	P	S	1
Collaborative Modelling			
Having everyone on the same page	P	S	18
Improved understandability	P	S	14
Pinpointing problems	P	S	13
Involving participants	P	S	5
Composing goal models should be more structured	N	U	2
Few feelings about collaboration	N	S	1
Sharing tasks well	P	S	1
Method Comparison			
Making notes	P	S	1
Modelling in Scrum	P	S	1
Participation			
Participation level is satisfactory	P	S	13
Too detailed discussions of system design are not interesting	N	S	4
Background with ICT helps to participate	N	S	2
Visual Representation			
Goal model representation - benefit	P	S	25

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 that interest exist to use new tools that are better tailored than the chosen free tools in the running case.

Next, we discuss findings related to AAOM shortcomings.

Table 6: Project setup and tooling themes

Theme name	**Theme**
Elaboration Sessions	How much time does it take to follow AAOM practices?
Modelling Time Usage	How long does it take to sketch system needs with AAOM?
Tools Usage	What is the effect of tooling on AAOM application?

4.3 Further AAOM-refinement needs (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. The gathered codes are depicted in Table 8. From 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 about concepts of quality goals, user stories and roles being clear.

On the negative side, we gather contradicting information if creating user stories for the lowest level goal in goal models is clear. Both codes have high ranking while statements about the process being unclear ranked a bit higher. Studying this contradiction is future research work in addition with 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.

Next, we discuss codes that emerge outside of research questions providing hindsight into the AAOM evaluation.

4.4 Emerged results

Emerged codes result from grounded coding that is explained in Section 3.4 and depicted in Tables 9 and 10. The participants express their feelings in the theme *Drawbacks* about the project setup that is not directly related to AAOM modelling, but 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 second top mentioned code. Also analyst is identified as carrying vital role in AAOM usage by having most responsibility in modelling activities. As a final statement worth noting, is a fact that doubt exists pertaining to the method suitability in smaller projects due to a pos-

Table 7: Project setup and tooling codes

Theme/Code	**Polarity**	**Type**	**Value**
Elaboration Sessions			
Session length suitable	P	S	5
Sessions could be earlier, less tired	P	U	2
Shorter sessions for inexperienced in analysis	E	U	2
New ideas since previous meeting offloading	P	S	1
One topic per meeting	P	S	1
Only new info on sessions	P	S	1
Modelling Time Usage			
Quickly to development	P	S	11
Refining goal models is fast	P	S	5
Method used before AAOM takes a lot of time	P	S	5
Time used effectively	P	S	4
From idea to user story in moderate time	N	S	1
Tools Usage			
Manual integration works but a lot of extra work	E	S	9
Good enough for starters	N	S	6
Need more integration	E	U	5
Commercial tools better	E	S	4
Commercial tools expensive	E	S	4
Dedicated tool support	E	S	2
Easy and flexible	P	S	2
Interesting to use new tools	P	S	1
New tools need training	N	S	1

Table 8: AAOM-refienement theme and codes

Theme/Code	Polarity	Type	Value
Method Clarification			
Sequence of activities clear	P	S	30
Goal model understandable	P	S	26
Usage of quality goals understandable	P	S	21
User story concept understandable	P	S	20
From goals to user stories unclear	E	S	16
From goals to user stories logical	P	S	10
Usage of roles clear	P	S	10
Quality goals link to user stories unclear	E	S	9
User story concept unclear	E	S	7
Development process unclear	E	S	2
User stories created by analyst unclear	E	S	2
Goal model lowest level finding unclear	E	S	1
Quality goals should have more details	E	U	1
Roles useful for user story creation	P	S	1
Usage of quality goals unclear	E	S	1

Table 9: Emerged themes

Theme name	Theme
Drawbacks	What negative impressions exist about a project setup?
Expectations	What are the expectations about AAOM benefits?
Modelling Suitability	What are the AAOM suitability issues in IT projects?
New Ideas	What proposals exist for method improvements, or for project-setup improvement?

sible 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 from the 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 small-scale project that employ agile software development. A case-study based research approach is instrumental for an evaluation. We apply the method in real-life setting for developing a mobile app from scratch. In a running case, the AAOM-method is used during the planning phase for requirements elicitation.

Interviews with project participants provide the most relevant input for the AAOM-method's evaluation. We perform an analysis of interviews to check different aspects of the AAOM application. Interviews are coded by searching answers to research questions, also new knowledge outside the scope of research questions is brought out. We employ additional coding techniques to organize data along themes and by apply a ranking formula for analysing codes to reveal confirming-, or contradicting findings to research questions.

The analysis of interviews shows that the AAOM method provides guidance to requirements elicitation for the project. The visual approach presents an intuitive way for both clients and the development team to perceive how user stories are connected to system goals and vice versa. The method is extensible, encourages collaboration and participation, and does not take much time.

The benefits of using the AAOM-method from a user perspective is evaluated positively. Benefits are a secure feeling for project direction, mutual communication and discovering new angles in requirements for the project. The visual representation provided by AAOM for requirements engineering is intuitively comprehensible. The same holds for gathered information about AAOM-procedures and the corresponding sequence of activities, goal modelling and usage.

Time spent on AAOM activities for gathering requirements is found to be adequate and even accelerates the requirements-engineering process. Overhead during that process is marginally low while refining and working with AAOM-models during later system-maintenance stages is considered fast. A gath-

ered low-scoring contradiction claims AAOM is not suitable for smaller projects. The ideal length of elaborations sessions is 1.5 hours and less for inexperienced developers. For projects on 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. Furthermore, the AAOM-method enforces communication via collaborative modelling, improving understandability, pinpointing problems and involving participants. Also the participation level for different roles governed by AAOM, matches expectations.

Table 10: Emerged themes codes

Theme/Code	Polarity	Type	Value
Drawbacks			
Experienced participants required for full potential	E	S	22
Better guide for analyst needed	E	S	14
Analyst has the most responsibility	N	S	5
Method might be overhead for smaller projects	N	S	5
Goal models too general, need more technical details	N	S	2
Initial user stories take time	N	S	2
Initial models need refinement	N	U	1
Starting from scratch should be more structured	E	U	1
Expectations			
Updates to models and user stories	P	S	8
Working results	N	S	5
Extensible implementation	N	S	4
Need more resources to accomplish goals	E	S	1
Modelling Suitability			
Clarifies what needs to be done	P	S	10
Organizing thoughts	P	S	6
Modelling fits into various project setups	P	S	5
Quality goals provide value for analyst and developer not client	N	S	1
New Ideas			
Link metrics to goals	N	U	2
Models can be used for system documentation	P	U	2
Quality goals holding technical details	P	U	2

Findings outside the scope of research questions is collected thanks to semi-structured interviews used in this case study. One of the discovered factors identified is AAOM not being intuitively understandable. Before using it effectively, longer experience in ICT is needed or additional guidance should be available. Also the analyst role has been identified as too heavy in applying the modelling method. Outcome of the project depends too much on the analyst. Another type of feedback gathered is expectations, covering working results and extensible implementation. New ideas cover linking models to financial data for measuring goal effectiveness, using models for system documentation and quantifying quality goals.

As a limitation for this paper, the results gathered during the running case study have limitations. Three participants out of four fill the client role and thus, the collected information is mostly based on the client-side opinion about the AAOM-method. Only two of the four participants have experience participating in software development processes and consequently, feedback also stems from inexperienced participant. The latter is also important since this shows how easy it is for laymen to adopt AAOM for practice. Another limitation is a lack of interview feedback by the participants with respect to methods for agile and requirements/engineering methods. This comparison promises to 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. On the other hand and based on a feedback from participants of the running case, an improvement of the AAOM-method must focus on the following aspects. The role of quality goals must be explained more effectively at an earlier stage. Since the analyst plays an important role during applying the AAOM-method, we need a deeper understanding of this essential role. Finally, we need to further verify case study validity by repeating similar research without the limitations of the current study.

Acknowledgement

We thank Mari-Ann Vellerand and Hando Rand for donating their time to participate in the running case of this paper. We thank Pekka Abrahmson and Dietmar Pfahl for providing constructive feedback on the case study setup.

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 Professional.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). The agile manifesto.

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

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

Cockburn, A. and Highsmith, J. (2001). Agile software development: The people factor. *Computer*, 34(11):131–133.

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. *Science of computer programming*, 20(1):3–50.

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

Gibbs, G. R. and Taylor, C. (2005). How and what to code. *Online QDA*.

Hull, E., Jackson, K., and Dick, J. (2005). *Requirements engineering*, volume 3. Springer.

I. Sommerville, P. S. (1997). *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc., London, 2nd edition.

Jensen, D. (2008). *Confirmability*, page 113. SAGE Publications, Inc., 0 edition.

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 Professional.

Leffingwell, D. (2013). Scaled agile framework. *Siehe: http://scaledagileframework. com*.

Leffingwell, D. and Aalto, J. (2009). A lean and scalable requirements information model for the agile enterprise. *Modern Analyst. com, posted*, 2.

Leffingwell, D. and Widrig, D. (2000). *Managing software requirements: a unified approach*. Addison-Wesley Professional.

Miller, T., Pedell, S., Sterling, L., and Lu, B. (2011). Engaging stakeholders with agent-oriented requirements modelling. In *Agent-Oriented Software Engineering XI*, pages 62–78. Springer.

Neill, C. J. and Laplante, P. A. (2003). Requirements engineering: The state of the practice. *IEEE Software*, 20(6):40–45.

Paetsch, F., Eberlein, A., and Maurer, F. (2003). Requirements engineering and agile software development. In *2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 308–308. IEEE Computer Society.

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

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

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131–164.

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

Saldaña, J. (2009). *The coding manual for qualitative researchers*. Los Angeles, CA [etc.]: 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.

Stake, R. E. and Savolainen, R. (1995). *The art of case study research*, volume 95004979. Sage publications Thousand Oaks, CA.

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

Sutherland, J., Viktorov, A., Blount, J., and Puntikov, N. (2007). Distributed scrum: Agile project management with outsourced development teams. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pages 274a–274a. IEEE.

Tenso, T. and Taveter, K. (2013). Requirements engineering with agent-oriented models.

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

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

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

Version One, I. (2015). 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 publications.