Agile Requirements Abstraction Model – Requirements Engineering in a Scrum Environment

Richard Berntsson Svensson¹, Sigurdur Örn Birgisson², Christian Hedin³, Björn Regnell⁴
Department of Computer Science, Lund University
Box 118
SE-22100 Lund, Sweden
0046 46 222 03 68
rbsv@cs.lth.se¹, cs03sb@student.lth.se², cs03ch@student.lth.se³, bjorn.regnell@cs.lth.se⁴

ABSTRACT
This article presents an investigation of the possibility to integrate traditional requirements engineering in an agile software development method. One reason for integrating a traditional with an agile method was that the case company experienced difficulties with the agile requirements engineering process. The results are based on a qualitative case study of one company that uses scrum for its software development. An agile model, with four abstraction levels was developed in close collaboration with industry. The model extends an existing abstraction model for traditional requirements engineering with aspects specific to agile processes. We conclude that it is possible to integrate traditional requirements engineering in a scrum environment and still stay agile in the development phase.

Categories and Subject Descriptors
D.2.9 [Software Engineering]: Management – software process models.

General Terms
Management

Keywords
Requirements abstraction, Product management, Product strategy, ARAM, Requirements engineering, Scrum, Agile

1. INTRODUCTION
Despite many efforts, software development has not been consistently successful [5]. Many projects have been delayed, failed, or rejected. Suggestions for improvement have come from experienced practitioners and a new class of software development process called agile methods has been developed [5]. The agile methods operate differently from traditional methods where the focus is on simplifying the software development process [8]. Agile methods share common principles, such as improved customer satisfaction, adapting to changing requirements, frequently delivering working software, and close collaboration between business people and developers [11]. Agile software development methods have become more popular and they have been successfully implemented in small and medium projects [8]. According to [2], in this decade the latest trend towards rapid application development and changing information technology has caused frustration among the traditional approaches that includes planning, specifications, and other documentation.

Requirements engineering (RE), on the other hand is a traditional method with the goal to identify, analyze, document and validate requirements [11]. Is it possible to bridge the traditional requirements engineering process with agile methods, and still stay agile? According to [11], requirements engineering and agile methods are seen as incompatible. One reason is that RE often relies on documentation while agile methods are focusing on face-to-face communication between customers and developers [11]. However, several studies [8], [9], [10], have looked into bridging traditional and agile methods. In [8], the study shows that it is possible to successfully integrate a traditional project management approach with XP [1] projects. Furthermore, Boehm and Turner [3] concluded that agile concepts will continue to integrate into traditional organizations and vice versa. In addition, Nawrocki et al. [10], proposed modifications to the XP methodology to embrace RE practices. One of the modifications was to have a RE phase in the beginning of the project that provides a wider perspective, and still manage to stay as lightweight as the original XP [10].

In this paper we investigate the possibility to embrace requirements engineering in a scrum [16] environment in one case company, namely Tactel. There were two main reasons for integrating traditional requirements engineering and agile software development. (1) In contract-driven development, customers have different requests of how the requirements engineering process should be handled. Some customers are familiar with the agile concept and want to stay agile, while others may not trust the agile requirements engineering process and prefers a more traditional process. (2) Tactel had several problems with regards to the agile requirements engineering process while using scrum as development method, the problems are reported in [7]. Based on these problems the Agile Requirements Abstraction Model (ARAM) [7] was developed. ARAM is an adaption of the Requirements Abstraction Model (RAM) [6]. This paper presents the ARAM, how it was created in close cooperation with industry and validated through feedback from professionals, both within the company and by external experts. Although the model was developed based on needs identified at Tactel, the objective was for ARAM to be generally usable.

The paper is structured as follows: Section 2 gives an introduction to the research context and presents related work. Section 3 presents the research methodology while an introduction to ARAM is presented in section 4. The evaluation results are presented in section 5 and section 6 gives a short summary of the main conclusions.
2. BACKGROUND AND RELATED WORK
This section gives a short description of the case company. In addition, related work regarding scrum, RAM, and integration of traditional approaches with agile methods is presented.

2.1 Case Study Description
Tactel develops mobile applications, providing solutions and consulting services to many of the world’s major network operators and mobile handset vendors. Examples of applications are GPS solutions, Mobile TV, Bluetooth solutions, and content clients. All projects are run differently as insourcing, outsourcing, or solution projects depending on the nature of the project and the wish of the customer. Within Tactel there are three project departments, each department has more than one project. The Mobical project belongs to one of the three departments. ARAM was developed and evaluated with close collaboration with the Mobical project.

Mobical is a solution for synchronizing and backing up personal mobile phone data over the air to a remote server. The personal data may be address book contacts, calendar events, or text messages. Mobical supports end customers that have multiple mobile devices, but want to have the data synchronized between them, and as a security measure for having backup if something happens to their mobile device. Mobical is mainly sold to network operators that want a customized version of the generic product. The generic product, Mobical.net, is a concept platform where the project handles user accounts, usually handed out for free. In addition, a lot of feedback on bugs and new feature requests are collected from the users by the Mobical project.

The Mobical project consists of 11 members with the roles of product owner, scrum master, developers, testers, and customer project leaders. The scrum methodology is used as the software development process by the Mobical project.

2.2 Related Work
Scrum is described as a process that accepts that the development process is unpredictable and has a do what it takes mentality [14]. According to [15], scrum starts with the assumptions that software development is complex and unpredictable to plan in advanced. Instead, scrum has a set of key principles for software development, such as: adaptability to changes for ensuring the best possible product to be produced, frequent builds that can be inspected, partitioning of work and team assignments into clean, low coupling partitions, and constant testing and documentation of product as it is built [14]. Scrum implements an iterative, incremental skeleton through three roles, (1) the product owner that represents the voice of the customer, (2) the team that is responsible for developing functionality, and (3) the scrum master that is responsible for the scrum team [16]. According to Schwaber [15], a scrum project starts with a vision of the system to be developed. The vision is transformed into requirements and inserted into the product backlog. The requirements in the product backlog are prioritized and divided into sprints. A sprint is initiated with a sprint planning meeting [15], where the product owner and the team discuss the purpose, meaning, and intentions with the highest prioritized requirements in the product backlog. The selected requirements for the coming sprint are put into another log, namely the sprint backlog. During the sprint, the team meets on daily scrum meetings to track work progress and schedule other meetings.

The Requirements Abstraction Model (RAM) [6], which ARAM is an adaption of, is an empirically developed model that structures and specifies elicited requirements in market-driven requirements engineering. RAM specifies a number of abstraction levels for structuring requirements. In [6], the number of abstraction levels is four, but they state that this is tailored for their specific case study and can be variable. The four abstraction levels are, the product level represents the most abstract requirements, feature level requirements explain what features should be included in the system, function level describes what the user is able to do with the system, and component level requirements are often as detailed as a beginning of a solution to the problems on function level. RAM has three specified action steps, specified (elicited) is the first sketch of an idea about what should be included in the system, placed (evaluated) means to determine abstraction level, and abstraction (work-up) means that new requirements are created with connections to the original ones, but with another abstraction.

There are several studies related to integrating agile software development with traditional approaches. In [8], a case study of two software system projects that have used extreme programming (XP) for software development within the stage-gate project management models was presented. The results show that it is possible to successfully integrate the project management model with XP. Furthermore, Boehm and Turner [3] studied management challenges when implementing agile processes in traditional development organizations. The study identified three categories of real and perceived barriers to integrating agile and traditional methods. The study concluded that agile concepts will continue to be integrated within traditional organizations and vice versa. In addition, a study [9] looked into the possibilities of mapping CMMI and scrum. The study concludes that a few adaptations on scrum, makes scrum more compliant with CMMI project management process areas. Nawrocki et al. [10] proposed three modifications to XP with regards to requirements engineering, for example, written documentation of requirements that is managed by a tester. The contribution of ARAM is set around integrating a traditional requirements engineering process (ARAM) with a scrum development process in an industrial setting.

3. RESEARCH METHODOLOGY
The research was carried out in cooperation between Lund University and Tactel. The study was carried out using an action research [13] approach. Action research aims to influence or change some aspects of the research focus. Furthermore, action research involves the improvement of: practice, the understanding of practitioners, and the situation in which the practice takes place [13]. In this research, we are involved in improving the practice of requirements engineering bridging the use of an agile method, namely scrum by introducing ARAM at Tactel. In addition, we improve the understanding of how practitioners use ARAM and its environment where the practice takes place. The general objectives of the research are:

1. Is it possible to integrate the traditional aspects of requirements engineering with the agile method of scrum?
2. Is it possible to be agile throughout contract and planning, while still maintaining good requirements engineering practices?

3. Can product quality in agile software development be increased by introducing a requirements engineering process?

Eleven interview subjects were chosen to represent the entire Mobical team to provide a complete picture. The different interviewed roles were: 2 customer project leaders, 1 product owner, 1 scrum master/project leader, 4 developers, and 3 testers. The study consists of the following three steps.

3.1 Step 1 – Problem Definition

Planning: Step 1 involved a brainstorming and planning meeting to plan the study. The interview instrument was designed with respect to the identified goals of interests for the study. To test the interview instrument, two pilot studies were carried out to adapt and improve the instrument. The used interview instrument is presented in [7].

Data Collection: The study collected data in three different ways, namely interviews, observations, and archive studies. The interviews were conducted using a semi-structured interview strategy [13]. All interviews were carried out by the second and third author. All eleven recorded interviews were carried out during a three week period and lasted for 60 minutes. Transcripts of all interviews were made in order to facilitate and improve the analysis process. Further knowledge was gained from observing the current work process. Observations were performed without participation by the second and third author. The observations took place in scrum standup meetings, scrum sprint planning meetings, and scrum retrospectives and demos. By attending these meetings, we were able to ask about feelings and attitudes in real situations, which is one of the main advantages with observations [13]. In addition, several documents were studied to collect information to give fast and overall knowledge about the past [13]. The studied documents were: sales and marketing documentation, contracts, project plans, RFQs, functional and system specifications, and the current scrum tool.

Analysis step 1: The content analysis [13] involved creating categories where interesting parts from the transcripts were added and discussed. The second and third author examined the categories from different perspectives and search for explicitly stated or concealed pros and cons with the current process. The result of this work is reported in [7].

3.2 Step 2 – Model Definition

The model definition was based on the input from step 1 and the Requirements Abstraction Model (RAM) [6]. Based on high-level goals, RAM, and related work (see Section 2), The Agile Requirements Abstraction Model (ARAM) was defined comprising a modification of the abstraction levels in RAM (see Section 4) without interfering the principles of the agile method scrum.

3.3 Step 3 – Model Validation

An evaluation of the model was carried out by assessing it in several steps within the Mobical team through meetings, workshop and the use of a questionnaire. In addition, an external evaluation with four experts through a questionnaire was conducted.

Internal validation: First, a discussion meeting with the testers to evaluate early derivation from their perspective was conducted. Another discussion meeting with the scrum master and product owner was held. The discussion was similar to the one with the testers and generated valuable suggestions for improvement of ARAM. In both discussion meetings, physical paper cards representing the process artifacts were used to illustrate the process flow. A third and final discussion meeting with people from project management at Mobical took place. The new and improved process was presented and discussed among the participants. An implementation workshop was held after satisfaction in all three discussion meetings. The objective of the workshop was to implement the new process in a real project within the Mobical team.

External validation: For a general evaluation, an external validation was conducted with four experts outside Tactel. The specialists were chosen according to their area of interests and expertise, including two scrum experts from the industry, a RAM expert from academia, and a configuration management expert from academia. All four experts were provided with the new process and a questionnaire.

3.4 Validity

In this section, we discuss the threats to validity in research projects presented in Wohlin et al. [18], and the measures taken in the presented study to increase validity.

Conclusion validity: The conclusion validity is concerned with the ability to draw correct conclusions. Each interview was done in one uninterrupted work session. Thus, answers were not influenced by internal discussions about the questions. The sampling techniques may pose a threat to the validity of the investigation. To minimize this misrepresentation, all participants in Mobical were selected.

Internal validity: This threat may have a negative effect on the casual relationship between treatment and outcome. As the evaluations of ARAM were performed with different interview subjects, they expressed their opinions and views regarding the current process and about ARAM. As their answers were recorded by the researcher this may have constrained people in their answers. Recorded answers were only to be used by the researcher, i.e. not to be showed or used by any other party. To avoid evaluation apprehension, complete anonymity from other participants was guaranteed.

External validity: The external validity is concerned with the ability to generalize the results, in this case the applicability of ARAM in industry at companies other than Tactel. Some of the problems introduced as a motivation behind ARAM could, to some extent be general for organizations faced with similar developing situations. However, it is not possible to generalize the results from this evaluation based on the case study of Tactel; although the concepts and the practical application of ARAM as described in this paper and in [7], makes it possible for any organization to adapt the concepts behind ARAM to fit their organization.
4. ARAM STRUCTURE

This section describes the ARAM’s structure. The version of ARAM presented in this paper is based on the development of ARAM at Tactel and an example of how ARAM can look like. The concepts of the ARAM model as described in this paper, makes it possible for any organization to adapt and tailor ARAM to fit their specific organization. The aim of ARAM is to bridge the traditional approaches and agile methods by allowing project plans and contacts, and still keep time-boxed iterations where the development team can work on their sprint backlog items. ARAM does not interfere with the scrum sprint, rather focus on making the pre-project requirements engineering efficient. One goal of ARAM is to produce a product backlog with testable and high quality requirements, which reflects customer needs. Customer involvement is important in agile aspects [12], [16], therefore, one path of ARAM relies on trusted customers that are willing to cooperate throughout the requirements engineering and development phase. However, customers sometimes find it difficult to understand or trust the agile requirements engineering process [4] and contracts. Therefore, a traditional path is available in ARAM, which fits into the process and enables the project team to be agile and using scrum.

Before the requirements engineering phase starts, a project inception phase needs to be conducted. During the project inception phase, the customer is informed about the two different paths that can be followed during the requirements engineering phase. The first path, the traditional path includes a software requirements specification and a project plan. The second path, the agile one involves a closer collaboration between the customer and software vendor. It is the customers’ decision if they want to follow the traditional or the agile path. Both paths are possible to use in contract-driven and market-driven software development. The requirements engineering phase using the model involves the following three steps: (1) elicitation, (2) placement, and (3) work-up. In addition, if the traditional path is chosen by the customer, a fourth step, (4) documentation is added, as can be seen in figure 1.

At the end of the requirements engineering phase (work-up for agile path and documentation for the traditional one), the requirements are inserted into the product backlog. The project inception phase as well as each of the requirements engineering steps is described in further details below. In addition, how requirements on different abstraction levels are used in scrum is also described.

4.1 Project Inception Phase

In a contract-driven business scenario, the customer contacts the software vendor and requests for product information. The response to the customer at this stage is an overview of the product containing the product’s feature and what value the product offers. We suggest that additional information, such as the agile development methods, is introduced in the product information response. This allows the customer to choose of participating in the agile process from the start (otherwise the traditional path is chosen), and in particular has a special type of contract written. The agile information should include an overview of scrum, target-cost contracts and how the agile path works.

The contract and requirements processes will ultimately end up in the product backlog, but there are two different paths to reach that goal. If market-driven development process is preferred, there is no direct customer to contact. In this case, the project inception may come from the product manager, marketing department, or from an innovation meeting at the producing company where high level goals are derived.

In the agile path, a target-cost contract with a variable scope is selected. The delivery date and the budget are known in advanced while requirements will be implemented based on their priorities. A release is specified where the most important features are delivered, which is the recommended process in scrum. Changes are allowed, and encouraged, throughout the development process, and new versions of working software will be released at the end of each iteration, allowing the customer to provide feedback. In the traditional path, a standard fixed-price contract is selected. The main advantage for the customer is that the contract states the total price and time of delivery. One disadvantage with fixed-price contracts is that the tender procedure is likely to favor the cheapest, or most desperate, alternative [12]. In addition, fixed-price contracts require that cost is estimated up front [7].

4.2 Elicitation Step

For customers who choose the agile path the next step is to cooperate with the software vendor to formulate a product vision statement, which contains overall business values and high-level requirements. In a market-driven development situation, a vision statement needs to be created. However, the vision statement is based on input from the product manager, market department, and market analysis. For customers who choose the traditional path submits a request for quote (RFQ) document to the software vendor. The RFQ is a tender proposal containing requested features and may be quite detailed, depending on the customer’s needs.

When the software vendor has received a vision statement or a RFQ, in order to get a uniform way of specifying requirements,
six attributes need to be specified in the elicitation step. The six attributes are:

Title: The title should reflect the contents of the requirement.

Description: The requirement description should describe the central characteristics of the requirement in a natural language. If additional information, such as use cases or screenshots, is available, it is stated in the description with a reference to the external location.

Value: This attribute consists of two parts: “why” the requirement exists and what “value” the specified requirement adds to the customer.

Source: Identifies the origin of the requirement, which in most cases, is from the RFQ, the vision statement, or another requirement ID it is worked up from. However, it may also be a change of requirement from the customer.

Owner: This attribute indicates who is responsible for the specified requirement, in the case of Mobical, most of the time it is the product manager.

Status: Keeps track of the requirement’s status. Based on [17], the suggested requirement statuses are: proposed, approved, rejected, implemented, and verified.

### 4.3 Placement Step
ARAM consists of a number of abstraction levels (based on the abstraction levels from RAM [6]). This step involves analyzing what abstraction level a requirement is on and place the requirement on its initial level. The abstraction levels are: product level, feature level, function level, and component level. Table 1 summarizes all abstraction levels with a short description linked to each. A more detailed description of the different abstraction levels can be found in [6], [7].

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Product level</td>
<td>Abstract and goal-like requirements</td>
</tr>
<tr>
<td>2</td>
<td>Feature level</td>
<td>Features that the product supports, but not too detailed</td>
</tr>
<tr>
<td>3</td>
<td>Function level</td>
<td>Functional requirements, actions that are possible to perform</td>
</tr>
<tr>
<td>4</td>
<td>Component level</td>
<td>Detailed description of how something should be solved</td>
</tr>
</tbody>
</table>

Deciding a requirements abstraction level may be challenging and requires additional analysis. The following question can be asked to decide a requirements abstraction level:

Q1. Is the requirement comparable to the product strategy? If yes, product level is the correct abstraction.

Q2. Does the requirement describe what the system should include/support? If yes, feature level is the correct abstraction.

Q3. Is the requirement functional and testable? If yes, functional level is the correct abstraction.

Q4. Does the requirement suggest how to solve something? If yes, component level is the correct abstraction.

In addition, figure 2 connects requirements at different abstraction levels to business plan, story repository (backlog) and different testing strategies to decide a requirements abstraction level.

![Figure 2. Abstraction levels connection](image)

For example, if a requirement needs to be tested at unit level, it is likely to belong at the component level. It is important to note that the business abstraction level is not part of ARAM (it is part of RAM, but it is recommended in the RAM implementation guidelines), but shown in figure 2 to reflect the fact that a business plan contains abstract high-level goals.

### 4.4 Work-up Step
Subsequent to the initial placement of the requirement on an appropriate abstraction level, the work-up of the requirement can begin. This means creating new requirements or link existing ones, which are connected to the initial requirement upwards or downwards in the abstraction level structure. The first work-up rule of ARAM states that all requirements must have a connection to the product level. This means there will be no loose requirements floating in the air. The second rule of ARAM is that requirements only need to be broken down to feature level. The reason behind the second rule is that elicited feature requirements may not have enough details at this stage for being broken down further. The work-up step in ARAM consists of three rules, which are:

Rule 1: All requirements should be abstracted upwards to reach product level.

Rule 2: All requirements should be broken down to reach feature level.

Rule 3: Broken links in the requirements abstraction chains must not exist.

We exemplify working up requirements by describing how the ID attribute is constructed. The ID attribute is composed in the form of a.b.c.d where each letter represents a link to a requirement. An example is illustrated in figure 3.
For the first requirement, the first available integer at abstraction level is used as ID. If the requirements repository is empty, integer 1 is chosen. The second requirement (abstraction level 2) gets the ID 1.1, the third one (abstraction level 3) gets ID 1.1.1, while the forth gets ID 1.1.1.1. The number of digits indicates the requirement’s abstraction level. The requirements exemplified above are all connected in the same structure.

If a new requirement is added to the same structure, it has to be analyzed and inserted into the same structure on the correct abstraction level. The new requirement’s correct abstraction level is at feature level and the parent (product level requirement) is “Sync over the air”. When the new requirement is inserted into the structure, the ID attribute is 1.2, hence the new structure is illustrated in figure 4.

It is possible to have requirements that are not connected to a parent requirement yet. For example, a detailed requirement at the component abstraction level, where requirements at higher levels are not yet identified. However, it must be connected to a requirement at the product abstraction level. The ID for the above described requirement could be 1.0.0.2, where zeros indicate that there is no requirement at those specific abstraction levels. However, rule 3 states no broken links must not exist, but the empty levels are only temporary while specifying the requirement with attributes and while working in the work-up step.

The goal of the work-up step is to have requirements that are suitable for inserting the product backlog and can be compared during prioritization.

### 4.5 Document Step
For customers who choose the agile path, the requirements engineering phase ends with the work-up step. It is not necessary to have any more documentation, as the agile customer is aware of what is in the product backlog. The customer prioritizes the requirements in the product backlog together with the product manager. For customers who choose the traditional path, the requirements are inserted into a software requirements specification (SRS). Typically, the abstraction levels are used for sections and categories, an example is illustrated in figure 5.

### 4.6 Scrum and ARAM
Requirements in the product backlog need to be prioritized according to most value for the customer. One main advantage by using ARAM is that all requirements can be compared. A requirement on feature level should only be compared with other requirements on the feature level. In other words, requirements should only be compared with requirements on the same abstraction level. After prioritizing all requirements in the product backlog, they are analyzed at the next sprint planning meeting. The high priority requirements are selected for the coming sprint. When estimating the effort for each story, requirements need to be broken down to at least the functional abstraction level to be able to make as accurate estimations as possible.

An agile practice and basic rule of scrum, is to develop a **common language** for all team members [7], which was pointed out as one problem in the case study [7]. Therefore, it is desirable to have common names for requirements at each abstraction level, as is illustrated in figure 6.

### 4.7 Process Overview and Summary
In the project inception phase, the potential customer contacts the vendor and request information (RFI) about the product. The
vendor replies with product information, where the two paths (traditional or agile) are presented to the customer.

In the traditional path (see figure 7), the customer presents a request for quote (RFQ), which contains a proposal based on the product information. Based on the RFQ, the requirements engineering phase starts (between RFQ and Product backlog in figure 7) where requirements are analyzed and worked up to the right abstraction level (as described in Sections 4.2 – 4.4). When the requirements are well-structured and of high quality, they are transformed into a software requirements specification (SRS). The SRS together with a project plan are sent back to the customer and a formal fixed-price contract is established. The requirements in the SRS are entered to the product backlog where they are prioritized and turned into a sprint backlog which the development team can begin to work with.

![Figure 7. Traditional process overview](image)

In the agile path (see figure 8), the customer follows the same initial steps as in the traditional path.

![Figure 8. Agile process overview](image)

After receiving product information the vendor and customer collaboratively construct a vision statement containing business needs and goals. At the same time, a variable-scope and target-cost contract is signed by both parties. The contract allows for continuous cooperation with requirements that are analyzed, worked up and down to the right abstraction level (between the vision statement and product backlog in figure 8), and inserted to the product backlog. Based on standard scrum procedure, the product backlog is prioritized and acts as a basis for the sprint backlog.

5. ARAM: EVALUATION

In this section, both the internal and the external evaluation results are presented.

5.1 Internal Evaluation

In general, all participants expressed a positive view of the ARAM. In particular, the participants appreciated the break down of requirements that are treatable and that the requirements are separated from the sales documentation. In addition, the connection between early project work and development was appreciated.

In the ARAM, one step is to place the requirement on an abstraction level. When first introduced to ARAM, several participants had expected a more difficult task than it turned out to be, due to a well defined process. By introducing ARAM, working with requirements engineering will be more time consuming than before. Some participants pointed out that they might have to wait with using the requirements process until the contract has been signed by the customer, to minimize unnecessary work in case the contract is not won. In most situations the customers sign the requirements as a part of the contracting, which in practice means two sets of requirements: the ones used to sign the contract and the ones extracted using the new requirements process. This is a threat that may affect managements’ willingness to implement the ARAM in real projects. However, it was pointed out that the Mobical projects often work in a similar manner; it may be possible to create a template, thus only have to do the major part of working up requirements and analyzing once.

All participants identified that ARAM makes requirements verifiable and improving the customer communication, which were two major problems before. However, one participant expressed that many problems are not only due to a lacking requirements process, but a lack of resources in general. This is a managerial problem and will not be addressed by the ARAM. In general, the participants believed that the new process will address the problems they had. In addition, the main benefit of using ARAM is the improved support for testing and, in the end, product quality according to the participants. Furthermore, ARAM leads to validated requirements that are easier to read, less ambiguous and less likely to contain errors. The benefit of improved prioritization was also pointed out by participants as a big reason to start to use the new process.

The main drawback with ARAM, which cannot be prevented, is that more time and resources need to be spent on requirements engineering. The participants, however, believes that the advantages, will cover these extra expenses. It was noted that customers may be unwilling to have requirements structured in the way ARAM propose, and may have their own template for requirement specifications. In this case, extra effort needs to be used on translating the proposed requirements structure to the customer’s format. In addition, the participants believed that project start up and planning projects will take more time, however, the extra time spent during project start up may be regained during the development according to the participants.
ARAM offers the customer to choose between two paths, the traditional and the agile one. Which path will Mobical’s customers most likely choose? Since Mobical’s customers are mostly major mobile phone operators that are quite rigid and formal in their purchasing procedures, the participants agreed on that most likely they will chose the traditional path. However, the product owner was open to the idea of trying to communicate the agile path, perhaps as a checkbox in the sales documentation. A project manager noted that it might be possible to convince some of the customers with long term relationships to use the agile path. However, the agile path will most likely require more resources from the customer, in particular if they are to participate during sprint demos etc.

5.2 External Evaluation
The general opinion among the external experts was that it is good to have multiple paths, due to that different customers wants different ways of working. The experts do not see any obvious contradictions with the two paths and how they work. The agile path was the most preferred one among the experts, which is in contrast to what is believed, among the practitioners, to be the most favorable path for their customers. One of the experts explained, if using agile development methods then I would prefer to continue to be agile. One expert describes that in the market-driven perspective, as in the context RAM was built, it would be easier to go the agile path. This would require a product manager that knows the market and what is expected by the end customers. Another expert liked the concepts of agile contracts, which is important to create the incentives with the customer to start the use of the agile path.

Is it possible to integrate traditional requirements engineering with scrum using ARAM? The experts do not see any contradictions between the two. ARAM teaches agile people to perform good and lean requirements engineering. One expert explained that ARAM realizes the traditional product management vs. project management. Where pre-scrum process being product management deriving requirements from a larger perspective and scrum being the project receiving them, and redefining into tasks and user stories for the programmers. However, product management needs the team when estimating. It is also expected to be good to be able to work on requirements in an iterative way, both pre- and in project as suggested by ARAM. There was one hesitant comment by one expert about working with really scared customers, where a better way to go would be a smaller pilot project instead of our pre-project activities, to minimize the risk of wasting resources.

The two academia experts did not see any constraints to the agile philosophy in the ARAM, while the industry experts were more reluctant. According to the industry experts, we are not doing anything that is not useful, no waste, and we are embracing change. One expert from academia explained the parallel to product and project management. What is waste from a project view does not have to be waste from a product perspective, meaning that documentation to some extent is necessary for the product to be delivered completely, while the project itself is not in need of this documentation. It is important to realize these differences, and that is one part that the industry experts maybe have not taken into account. In addition, having good people in the organization is more important than introducing the ARAM.

However, one expert lacks the agile approach to prioritization in the ARAM, and that ARAM creates too much of waste according to what is needed in the beginning of the project, not enabling the prioritization to be done with only user values. The expert continues, this may be the compromise that is needed to stay agile.

ARAM is overall seen as an easy to use process by the experts, but with an obvious initial learning curve and initial sacrifices in velocity. Furthermore, ARAM is a reasonable way of working, no more or less than what is required to get the job done anyway. One expert describes the needs for easy to use support manuals and tools to use within the working process. In addition, thorough static and dynamic validations in the industry are needed before the question really can be answered. Another expert likes the good description and support questions presented of the abstraction levels, since this is the uttermest tool in the process.

There were a few problems with ARAM identified by the experts. The first problem regards configuration management. Is sprint planning different from a change control board meeting in a more lean way where the product owner is present? Change requests are handled in agile methods, but without the involvement of money. The drawbacks on implementing changes are instead of money, the number of planned requirements, which are reduced if a change is accepted. By not involving money, change management is leaner. The second identified problem is related to requirements prioritization, it may be a little too late in ARAM. A third issue was the lack of dependency analysis in ARAM, for improved transparency and awareness within the project. In addition, there could be biases involved when creating the different levels of requirements, people creating more “selling” requirements for the ones they feel is more important. The experts suggest at least creating an awareness of this, and maybe including some kind of review of this activity.

The experts ask for a more thorough evaluation of the ARAM to be able to evaluate its adaptability in general. The external and internal evaluations are a first steps towards, but further empirical evaluations are needed.

6. CONCLUSIONS
The Agile Requirements Abstraction Model was developed based on RAM in close collaboration with an industrial partner, Tactel. The goal was to integrate traditional requirements engineering in a scrum environment, and still be agile throughout the software development process. ARAM consists of two different paths, the traditional and the agile path and it is the customer’s choice of which path that should be used in the requirements engineering phase. ARAM follows a number of steps, regardless of chosen path. The first three steps are elicitation, placement, and work-up. A forth step is added in the traditional path, namely documentation. ARAM enforces work-up rules that abstracts and breaks down requirements as needed. Product level requirements can be compared to goal-like requirements. Function and component level requirements can be compared to actions that can be performed and how it can be solved respectively.

As part of its development, ARAM was validated in industry through both internal validations, getting feedback from the Mobical team and through external validation, getting feedback from experts outside Tactel. The overall result indicates that
ARAM is relevant in integrating requirements engineering in a scrum environment. During the validations it was discovered that the work-up step, placing requirements on the correct abstraction level, was not a difficult task. In addition, the benefit of improved prioritization was pointed out by the participants. Furthermore, the conducted evaluation shows that ARAM is an easy to use process by the external experts. The two main identified challenges were, (1) more time and resources need to be spent on requirements engineering, however, the Mobical team believes the advantages that ARAM brings covers the extra expenses, and (2) prioritization may be a little too late in ARAM.

The evaluation indicates that ARAM is feasible and relevant to the selected domain. We also believe that the general concepts of ARAM are transferable to integrating requirements engineering in a scrum environment for other domains, but this needs to be investigated in further research. Further research also includes, an additional evaluation of the ARAM when the model has been used throughout an entire project.

7. ACKNOWLEDGMENTS
We would like to thank all of the participants at Tactel who have helped in making the data collection possible for this research. Furthermore, we would like to thank all the external reviewers of ARAM for their valuable comments.

8. REFERENCES