

TALLINN UNIVERSITY OF TECHNOLOGY

Faculty of Information Technology

Department of Informatics

Chair of Software Engineering

MOBILE APP DEVELOPMENT USING AGILE AGENT-ORIENTED MODELLING

Master Thesis

Author: Irina Vorontsova

Student code: 040685 IAPM

Supervisor: Alexander Horst Norta,

Tanel Tenso

Tallinn

2015

Author declaration

I hereby declare that this thesis is based on my own work. All ideas, major views and data from different sources by other authors were only used as reference and/or for research purposes. The thesis has not been submitted for any degree or examination in any other university.

Irina Vorontsova

Date

Signature

Abstract

This work is devoted to the problem of evaluating a novel Agile Agent-Oriented Modelling(AAOM) method for engineering requirements in an agile software development process.

Currently, the use of agile methods during software development is a standard practice, and user stories is one of the most popular practices of breaking complex system requirements into smaller pieces.

However, user stories alone are not helping to understand a bigger picture of the system goals. There are some methods that try to solve this problem, but according to our experience most of them lack visual component and are often too heavy for smaller projects.

The AAOM method was developed to address this gap by adding a visual approach to agile requirements engineering that links goal-model creation techniques taken from agent-oriented modelling and connects them intuitively to user stories.

The case study based evaluation of this master thesis proves the applicability of AAOM for requirements engineering in an agile software development process. The case of study is development of mobile application to connect lost and found items with their owners.

The thesis is in English and contains 46 pages of text, 6 chapters, 6 figures, 16 tables.

Annotatsioon

Käesolev lõputöö tegeleb Agiilse Agent-Orienteeritud Modelleerimise(AAOM) meetoodika kasutamise evalueerimisega nõuete kogumiseks. AAOM uudne meetoodika on mõeldud kiireks nõuete selgitamiseks agiilse lähenemisega tarkvara arendusprotsessides.

Agiilsete meetoodikate kasutus tarkvara arenduses on tänapäeval kujunenud standardiks. Kasutuslood (User Stories) on agiilsetes praktikates üks kõige populaarsemaid vahendeid, et tükeldada suuri ja keerulisi süsteeme väiksemateks tükkideks.

Samas kasutuslugudest üksinda ei piisa suurema pildi saamiseks süsteemi eesmärkidest. Selle puuduse kompenseerimiseks on välja töödatud mõningaid meetodeid. Meie kogemusel enamustel neil puudub aga visuaalne esitus ja tihti on nad liiga rasked väiksemate projektide jaoks.

AAOM meetod on välja töödatud selleks, et lahendada täpselt neid kahte probleemi lisades lihtsa ja visuaalse lähenemise agiilsele nõudmiste kogunemisele. Meetod proovib intuiivselt ühendada eesmärgipuude modeleerimise agent-orienteeritud modelleerimise raamistikust ja kasutuslood agiilsest tarkvara arendusest.

AAOM meetodi rakendatavus tõestatakse käesolevas magistritöös juhtumiuuringu meetodi abil. Uuritavaks juhtumiks on agiilse lähenemisega loodud mobiilirakendus mille abil saab kaotatud ja leitud esemeid kokku viia nende omanikega.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 46 leheküljel, 6 peatükki, 6 joonist, 16 tabelit.

Glossary

RE – *RequirementsEngineering*

XP – *ExtremeProgramming*

SAFe – *ScaledAgileFramework*

AOM – *Agent–OrientedModeling*

AAOM – *AgileAgent–OrientedModeling*

LOC – *LinesOfCode*

Contents

1	Introduction	10
1.1	Requirements Engineering	10
1.2	Cornerstones	10
1.2.1	Agile	10
1.2.2	Requirements engineering in agile	11
1.3	Gap detection	12
1.4	Agile Agent-Oriented Modelling	13
1.5	Research questions	14
1.6	Research framework	15
1.7	Thesis structure	15
2	Research Background	16
2.1	Introduction	16
2.2	AAOM method explanation	16
2.2.1	Description	16
2.2.2	Notations	16
2.2.3	Approach for creating goal models	17
2.3	Case: Lost&Found mobile app project	18
2.3.1	Idea and goals	18
2.3.2	Lost&Found app concept	19
2.3.3	Project setup	19
2.3.4	App implementation details	22
2.3.5	Results	24
3	Case study research method adaptation	25
3.1	Introduction	25
3.2	Case study design	25
3.3	Data sources	26
3.4	Analysis procedure	27
4	Data collection: interviews	28
4.1	Introduction	28
4.2	Planning the interviews	28
4.2.1	Interview questions	29
4.3	Interviews conduction	29
4.4	Preparing the interviews for analysis	29
4.5	Coding the interviews	30

4.5.1	What is the coding	30
4.5.2	Codes	30
5	Interviews analysis	32
5.1	Introduction	32
5.2	Interviews' codes analysis	32
5.2.1	Codes attributes	32
5.2.2	Formula - finding a value of the code	33
5.3	Analysis of interviews codes by theme	34
5.3.1	Benefits	34
5.3.2	Collaborative Modelling	34
5.3.3	Drawbacks	35
5.3.4	Elaboration Sessions	35
5.3.5	Expectations	36
5.3.6	Method Clarification	36
5.3.7	Method Comparison	37
5.3.8	Modelling Suitability	38
5.3.9	New Ideas	38
5.3.10	Participation	39
5.3.11	Time Taken for Modelling Activities	39
5.3.12	Tools Usage	40
5.3.13	Visual Representation	40
5.4	Analysis summary	41
5.4.1	Gathered data limitations	41
5.4.2	Results	41
5.4.3	Method improvements	43
6	Conclusion	44
6.1	Summary	44
6.2	Answers to research questions	44
6.3	Limitations, open issues, future work	45
A	Appendix - Interview Questions	48
A.1	Client questions	48
A.2	Analyst questions	49
A.3	Developer questions	50
B	Appendix - Codes and Formula	51

C	Appendix - Design concept for Lost&Found mobile app	54
D	Appendix - Goal models for Lost&Found mobile app	55
D.1	Main goal model	55
D.2	Goal model for lost items	56
D.3	Goal model for lost animals	57
D.4	Goal model for lost people	57
D.5	Goal model for giveaways	58
D.6	Goal model for pre-registered items	58
D.7	Goal model for venue management	59
D.8	Goal model for revenue collection	59
D.9	Goal model for user handling	60

List of Figures

1	Example of goal model	13
2	Example of goal model with user stories attached	14
3	The notation for goal model	17
4	Trello scrum board	21
5	Draw.io diagram for goal models	21
6	Bitbucket wiki for the Lost&Found project	22

List of Tables

1	Mobile permissions required by the Lost&Found app	22
2	Predefined coding themes	31
3	Grounded coding themes	31
4	Codes for theme "Benefits"	34
5	Codes for theme "Collaborative Modelling"	34
6	Codes for theme "Drawbacks"	35
7	Codes for theme "Elaboration Sessions"	36
8	Codes for theme "Expectations"	36
9	Codes for theme "Method Clarification"	37
10	Codes for theme "Method Comparison"	37
11	Codes for theme "Modelling Suitability"	38
12	Codes for theme "New Ideas"	38
13	Codes for theme "Participation"	39
14	Codes for theme "Time Taken for Modelling Activities"	39
15	Codes for theme "Tools Usage"	40
16	Codes for theme "Visual Representation"	40

1 Introduction

1.1 Requirements Engineering

Requirements engineering (RE) is an important software development activity and usually one of the first phases. It is a process of formulating, documenting and managing the requirements for software [31, 15]. It consist of [31]: requirements identification, analysis, documentation and validation.

Often, clients are not sure about what they want or need, and the RE process helps to translate the client's unclear abstract ideas into precise and complete specifications understandable and implementable by developers. Too late detected errors in the RE-phase can be very costly [6], or might culminate in incorrectly running software when undetected at all.

Requirements engineering in agile software development methods will be discussed in this chapter, as agile is the most standard way of developing software today [1].

1.2 Cornerstones

1.2.1 Agile

Nowadays, software release life cycles are shortened because of a quickly changing requirements [8], and waterfall-like development methods are not suitable any more for such a changing environment. An iterative approach for building and delivering small parts of software incrementally comes in as a solution for rapidly-changing requirements: Agile [11, 4].

Agile software development is a group of software development methods, which are characterized by [26]:

1. Time-boxed iterative and incremental development
2. Frequent delivery of usable software
3. Collaboration with customer
4. Teamwork, self-organizing and cross-functional teams
5. Ability to quickly respond to changes

With an ability to ease adaptation to changing requirements, agile is widely used in software development projects by 88% [1] of developers.

1.2.2 Requirements engineering in agile

The RE-process in agile can vary a bit based on the chosen framework (Scrum, XP, Lean, Kanban), but there is a common list of practices used for managing RE [4, 26, 17, 3]:

1. Face-to-face communication over written specifications
2. Iterative RE
3. Requirement prioritization goes extreme (continual re-prioritization and business value driven prioritization)
4. Managing requirements change through constant planning
5. Prototyping
6. Use review meetings and acceptance tests

Writing user stories is one of the most popular and simplest techniques used in agile software development for requirements engineering [9, 25].

A user story is a written sentence or two that describes a desired functionality from the system's user point of view.

There are several formats and concepts for agile user stories, while for this research a format proposed by [9] is used:

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

The last part "so that <benefit>" can be omitted 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 to be implemented within one iteration. Larger user stories should be separated into smaller user stories [9].

1.3 Gap detection

There are some common problems within RE in agile development [26]:

1. Problems with accurate cost- and schedule estimation
2. Inadequate or inappropriate architecture
3. Neglect of non-functional requirements
4. Customer access and participation
5. Prioritization on a single dimension (business value only)
6. Inadequate or a lack of requirements verification (agile RE focuses more on requirements validation)
7. Minimal documentation

While using user stories has mitigated some of the problems listed above, there are further shortcomings [9]. For example, breaking up user stories into smaller pieces helps to organize tasks, while one of the limitations is that it is hard to get a bigger picture.

There are methods to mitigate these problems, namely Scaled Agile Framework(SAFe) [20] and a lean approach to agile requirements [19, 21], but these approaches are not really visual and, according to our experience, not graspable from the start by client. Furthermore, these methods are meant for large enterprises and 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 that are developed exactly for depicting system goals in a visual way. Goal based requirements engineering has been known for a while [15, 10, 34, 35] and are concentrating on eliciting requirements based on the use of goals that need to be achieved by the target system. One of the examples of goal modelling techniques is described in agent-oriented modelling(AOM) [32]. In addition to functional goals, AOM-goal models also include roles and quality goals (explained in Chapter 2.2.2).

Similar to SAFe and approach by Leffingwell [21] an idea of goal models is to provide hierarchical view of tasks for solving complex problems. But the benefit of goal models in front of preceding methods is simpler layout and visual composition [32, 10]. Knowing this advantage we came to an idea of combining agile development requirements artifacts via AOM goal models. We call this method Agile Agent-Oriented Modelling (AAOM) [33] and intend to find out whether this method is as sufficient as established methods

while being more lightweight.

1.4 Agile Agent-Oriented Modelling

The AAOM method is based on the Agent-oriented modelling(AOM) technique. The AOM is a holistic method for analyzing and designing socio-technical systems consisting of humans and technical components, as offered by [32].

We choose AOM because the models used in AOM are intuitively understandable for any stakeholder, even nontechnical people [32]. Goal models were chosen from among of all other AOM models to be used in the AAOM's RE approach because goals and quality goals on those models are representing functional and non-functional requirements of the system. Goal models also define roles and relationships between roles and goals/quality goals.

Goal model's example is presented in Figure 1.

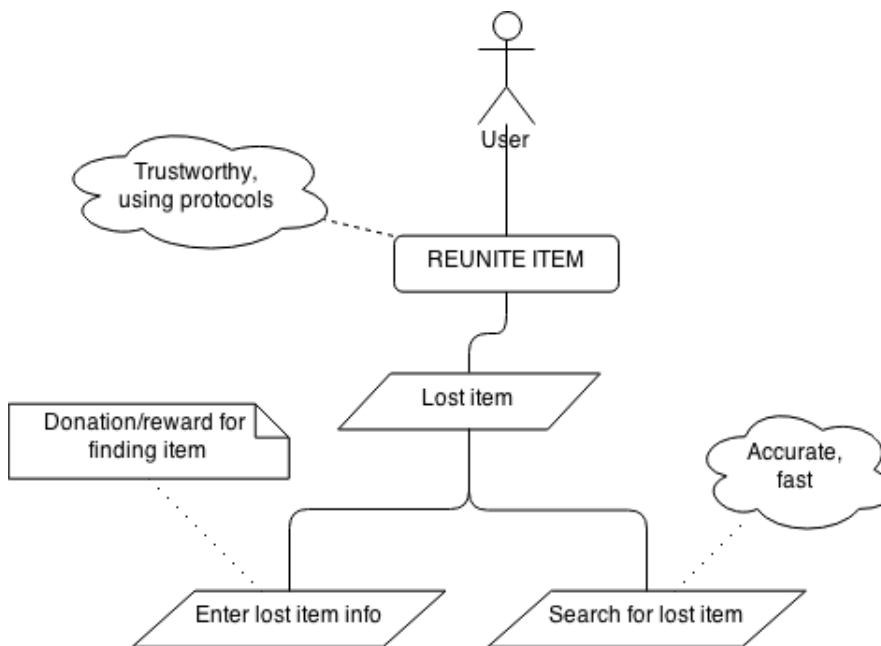


Figure 1: Example of goal model

In the AAOM method user stories are attached to lowest level sub-goals (Figure 2) in that way so that it allows to trace how every user story is connected to system's top goal.

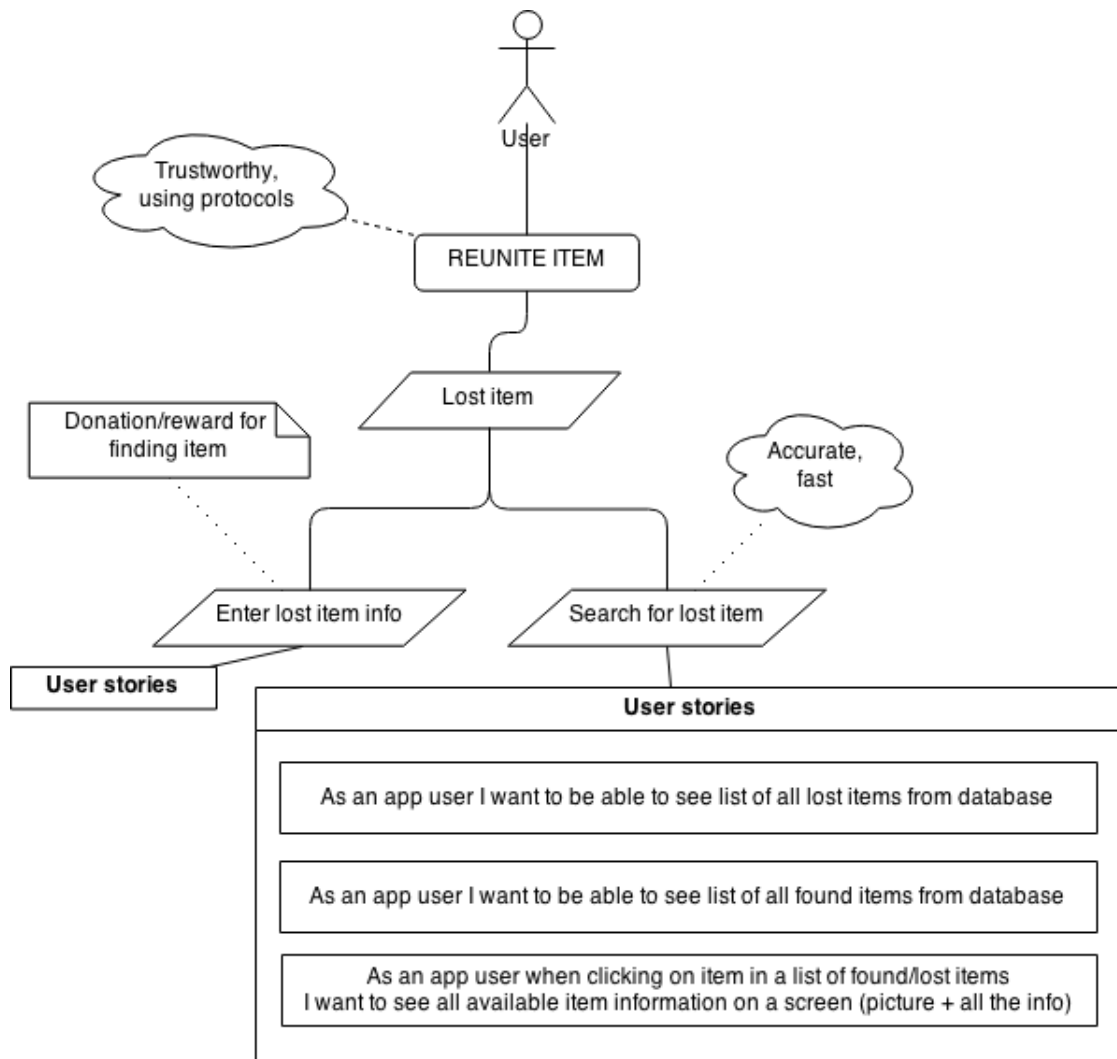


Figure 2: Example of goal model with user stories attached

All concepts of the AAOM and approach of goal model's creation are described in details in Chapter 2.2.

1.5 Research questions

The main research question of the current master thesis is: how suitable is AAOM for RE in agile software development? The main question is divided into next set of sub-questions:

1. How to evaluate AAOM with a suitable research method?
2. How to collect data for an AAOM-evaluation?
3. How to analyze the collected data?

1.6 Research framework

In order to answer research questions, a research methodology should be chosen. For evaluating of the AAOM method two options were available: an experiment or a case study.

An experiment requires to have a controlled environment where to perform checks and compare the results of manipulating one or more variables [27]. In case of evaluating the AAOM method for RE we need to find an agile software development project that agrees to interferences from researchers in project's RE phases, or to create an artificial project, maybe even isolate only RE process for studying. While it is already quite hard to find this kind of project, and creation of one would require too much effort, we are more interested in how the method would be used in a more realistic environment. That is where the case study research method suits the most.

Case study in software development implies that the method or technology will be observed and investigated in a real-life settings on a specific demonstration case [29, 28]. A case might be a specific project, or a company, or a concrete team, where the method or technology will be applied.

The case study approach and design used for this master thesis is described in Chapter 3.

1.7 Thesis structure

In Chapter 2, the Agile agent-oriented modelling method is introduced in details and the case, project for mobile app development Lost&Found, is presented.

In Chapter 3, case study research method is introduced together with design, data sources and analysis description.

Chapter 4 contains a description of the data collection stage of a case study, we are concentrating on the interviews as a main source of data.

In Chapter 5, interviews analysis is performed and results of the analysis are presented.

Finally, Chapter 6 concludes the master thesis results and gives open issues for future work.

2 Research Background

2.1 Introduction

In order to proceed with the given case study, the objective of the research and a running case are introduced and explained in this chapter.

The AAOM method is the object of current research and is aimed at mitigating the problem of getting overview of the project's main goals for all stakeholders in the RE phase in an agile development environment. It is based on goal models taken from agent-oriented modelling and connects them to user stories, artifact from agile RE.

The Lost&Found mobile app development project is a running case where the AAOM method is applied during the iterative RE phases. Project aim is to create a mobile app to help people to find lost items and provide an easy way to report findings.

2.2 AAOM method explanation

2.2.1 Description

The AAOM method is a technique for collecting and documenting requirements and at the same time resolving the problem of comprehending primary system goals in an agile software development environment [33]. The approach is based on using a goal modelling technique from AOM for requirements representation in an easy and visual way so that every stakeholder could understand it. The contribution of the approach lies in connecting goal models to user stories(Chapter 1.2.2).

2.2.2 Notations

Notations from Figure 3 are used in goal models of the agent-oriented modelling approach and are reused for AAOM goal models.






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

Figure 3: The notation for goal model

2.2.3 Approach for creating goal models

The creation of goal models in AAOM was described in [33] but enhanced by omitting behavioral scenarios. The modified guidelines are as follows:

1. Create the top-level hierarchy for the goal model:
 - 1.1. Define the main purpose of the system being developed. Introduce the purpose as the root goal.
 - 1.2. Expand the main goal into sub-goals. Each sub-goal represents one aspect of reaching the main goal.
 - 1.3. Where applicable, supplement the main goal and its sub-goals with quality goals. Quality goals represent quality aspects of the functional goal.
2. Expand the top-level goal model into lower level goals:
 - 2.1. Handle each sub-goal of the top-level goal model as the main goal.
 - 2.2. Apply Step 1 to each sub-goal.
3. Repeat expanding the goal model until reaching the lowest level of achievable and justifiable goals:
 - 3.1. The lowest level goal model is a goal that can either can be implemented by a single role, or if implementation of the goal can be easily described.

- 3.2. Improve goal models whenever more information becomes available.
4. Create user stories for the lowest level of goal models:
 - 4.1. A user story includes one aspect of the goal.
 - 4.2. The process of mapping goal to user stories goes as following:
 - 4.2.1. A user story is written from the perspective of a particular **role**.
 - 4.2.2. The **goal aspect** is an activity that represents how the corresponding goal is implemented. Multiple user stories can be used to describe how to complete a goal aspect.
 - 4.3. While creating the goal models, remember the following notes:
 - 4.3.1. All goal models are not required to be completed at once. Branches of goal model can be elaborated one by one in different iterations of an agile development.
 - 4.3.2. Goal models can be modified based on the data received during the implementation of user stories.
 - 4.3.3. Goal models demonstrate an overall view of the system and explain what needs to be accomplished. User stories demonstrate design details of a system and link goals to implementable features of the system.
 - 4.3.4. There is no limit for the number of user stories to be created for the lowest goal.

2.3 Case: Lost&Found mobile app project

2.3.1 Idea and goals

People tend to lose their belongings all the time due to forgetfulness or distraction. It is not so easy to find lost item, unless you know where to search for. Airports, taxies, cinemas might have their own lost items section, where you can turn to in order to find lost umbrella or gloves, or maybe even phone or wallet. But where do you go if you are not sure where exactly you lost your belongings?

Some sites exist, that might help you: in Estonia there are police site with listing of items brought to police by anyone, and lostnf.com which also lists items lost in trains and some taxi companies. Furthermore, there are some forums and facebook groups for reporting

lost and found items, but at the moment there are no mobile solutions for finding lost things.

The Lost&Found mobile app project idea is to unite people who lost something with items finders. The mobile app would be a simple and quick mobile solution to report findings by using smartphone capabilities: a phone camera allows to instantly make a picture of found item and location of an item can be read from smartphone GPS.

The app is also beneficial for the people who lost something: app allows to announce the loss and get notifications when item with similar description and attributes is found and entered into the database.

2.3.2 Lost&Found app concept

The mobile app has multiple goals. The main purpose is of course reuniting people who lost things with item finders.

However the app is not only for lost items, it is also for lost pets, maybe even cars and lost people, of course there are differences in the process of getting lost things back to the owner.

The app's additional functionality is a possibility to pre-register items before those are lost by using stickers with unique L&F code connected to the owner. Stickers would help a finder to identify an item's legitimate owner and contact him for a return.

Another possibility of an app is a giveaway section, which is meant for found items that no-one came back for. This section would allow to give up those owner-less findings for free to those in need of this item.

The Lost&Found app design concept can be viewed in Appendix C.

Goal models created for the Lost&Found app goals can be found in Appendix D.

2.3.3 Project setup

Members

Four people participated in this project, three of them were clients and one person played the role of analyst and a developer. We are paying attention to participant's experience in later analysis section. Person is considered experienced if he or she has at least one year of experience in his/her role. One of the clients is considered as an experienced client

and two of them not so experienced. The developer is experienced and the analyst not so experienced.

Process management

An agile software development was used for creating the app prototype. As there was just one developer implementing mobile app prototype, no complicated processes were needed. More specifically, some scrum techniques were used: planning with user stories, backlog created, short iterations: we had three iterations overall, work visualizations with scrum board, meetings with clients after each iteration end. The AAOM method was used for planning sessions.

Tools

Following free online tools were used to support the Lost&Found project software development:

Trello¹ - a collaboration tool to organize project tasks into board. Helps to visualize the tasks and progress of software development.

The Lost&Found Trello board (Figure 4) was organized into next categories:

- Backlog - whole backlog of tasks for implementation of all app
- TODO - tasks to be done next
- Doing - tasks in progress
- Done - already implemented and finished tasks during a current iteration
- DoneDone - tasks finished during previous iterations

¹<https://trello.com>

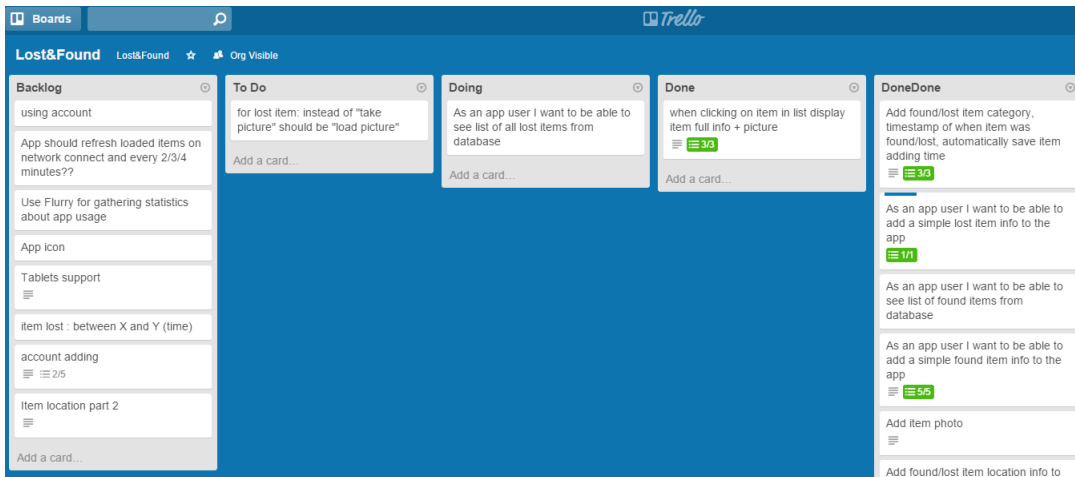


Figure 4: Trello scrum board

Draw.io² - online diagramming tool for creating charts and diagrams. It was used for drawing goal models(Figure 5) for Lost&Found project.

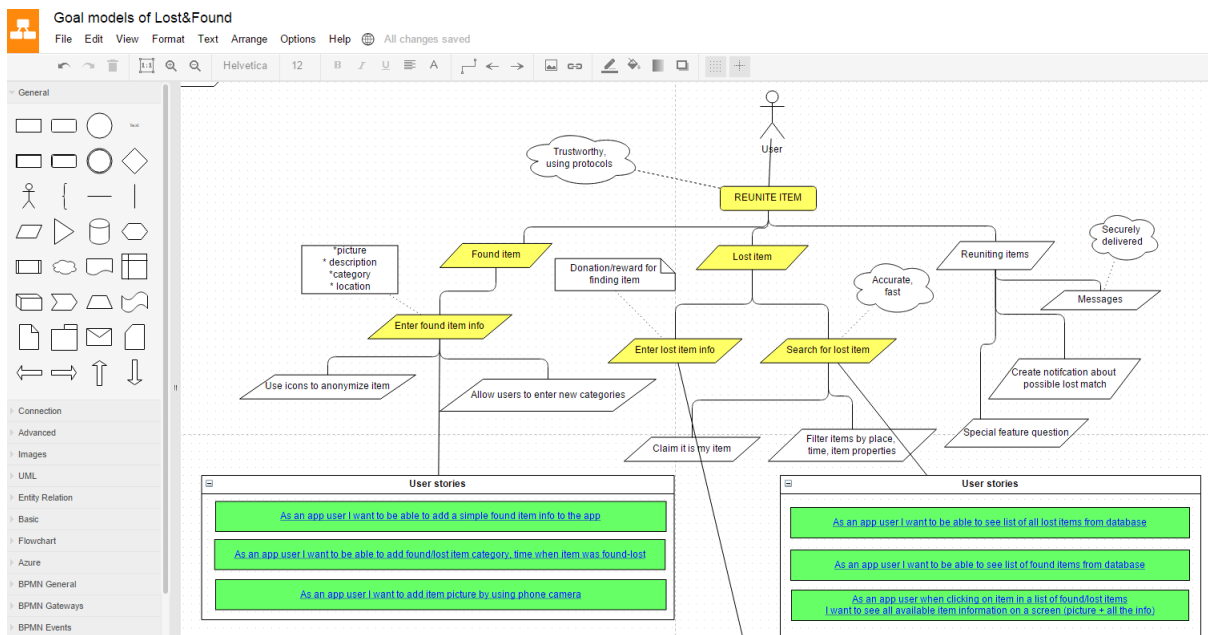


Figure 5: Draw.io diagram for goal models

BitBucket³ - free source code hosting for Git and at the same time simple wiki(Figure 6) and issue manager for the project.

²<https://www.draw.io>

³<https://bitbucket.org>

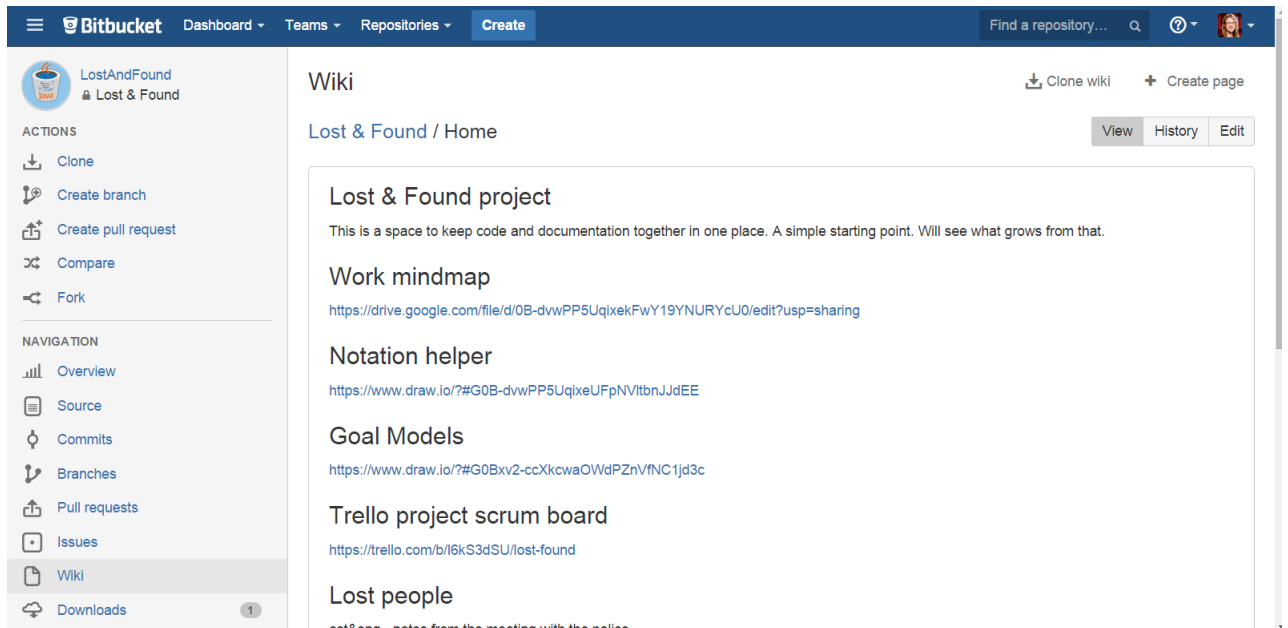


Figure 6: Bitbucket wiki for the Lost&Found project

2.3.4 App implementation details

The Lost&Found project ultimate aim would be to have mobile apps for all major mobile platforms i.e. iPhone, Android, Windows Phone.

Device permissions, required by the Lost&Found app, are listed in Table 1.

Table 1: Mobile permissions required by the Lost&Found app

Permission	Purpose
Network communication	To connect to external databases of lost and found items
Location	To determine found item locations
Camera	To take a photo of found items
Phone storage	To save photos of found items on the phone

To achieve the best quality and user friendliness in terms of UX(User Experience), it was decided to go for native mobile development and to implement first mobile prototype for Android platform.

These technical choices were made for the Lost&Found mobile app development:

- Android versions 3.0 - 4.4.2 (Android API 11-19) are supported by the app, so that 81%⁴ of all existing Android devices can be covered.

⁴<http://developer.android.com/about/dashboards/index.html#Platform>, data for year 2013

- MongoDB⁵ NoSQL document database is used as a back-end database and MongoLab⁶ hosting platform is used as a database external provider that allowed us to use free 0.5 GB storage, just enough for prototyping database.
- Maven⁷ project management tool is used for project's build.
- Git⁸ is used as a free and open source version control system - because we decided to use BitBucket as a source code hosting.

⁵<http://www.mongodb.org>

⁶<https://mongolab.com/>

⁷<http://maven.apache.org/>

⁸<http://git-scm.com/>

2.3.5 Results

The team had three iterations that followed the planning sessions. After three iterations the Lost&Found app prototype with minimal required functions was ready. The prototype implementation followed "Lost items" goal model from Appendix D.2.

The app prototype is connected to external database that holds the list of lost and found items entered by users and displays that list of items to the user if it has access to network. It is possible to add found or lost items by entering item details: description, category, the date of finding/losing, location, picture.

One may add an item picture by using phone camera. Item location is determined by phone location.

Following user stories were fully implemented for the Lost&Found app prototype:

1. As an app user, I want to be able to add a simple found item info to the app
2. As an app user, I want to be able to add a found/lost item category and time when item was found-lost
3. As an app user, I want to add an item's picture by using a phone's camera
4. As an app user, I want to be able to add a simple lost item info to the app
5. As an app user, I want to be able to add found/lost item locations by using phone location
6. As an app user, I want to be able to see the list of all lost items from a database
7. As an app user, I want to be able to see the list of all found items from a database
8. As an app user, when clicking on an item in a list of found/lost items I want to see all available item information on a screen (picture + all the info)

Now, when the demonstration case and the AAOM method as an object of the case study are explained, we can continue with designing the case study.

3 Case study research method adaptation

3.1 Introduction

As already mentioned in the introduction, a case study research method was chosen to conduct the evaluation of the AAOM method. A case study in software development is a research to investigate and observe a new method or technology in a real-life settings in a specific demonstration precedent (case) [29, 28]. It is also specified that object is better studied by looking at it from multiple perspectives [27].

A case study consists of next stages: case study design, preparation for data collection, collecting evidence, analysis of collected data and reporting [29]. All of those stages were conducted in current research to a greater or lesser extent.

Case study design helps to understand how to assess the AAOM, and evaluation points are described in Chapter 3.2. Sources of evidence for this study are discussed in Chapter 3.3 and analysis procedure in Chapter 3.4.

3.2 Case study design

The object of a study is a new method to gather requirements: AAOM, which is applied to an agile development environment.

A detailed investigation in a context of one project, or a single-case design, is used. The case under observation is required to be a software development project and to be based on iterative requirements gathering. It was desirable that the project would have several different goals and complex requirements. The Lost&Found mobile app development project, described in details in Chapter 2.3, is the demonstration case.

The case study investigation questions together with achieving techniques were defined as follows:

1. Investigate if the visual approach is understandable via interviews.
2. Measure time and use interviews to detect time used for modelling activities.
3. Use interviews to find the participating level/willingness to participate in requirements elicitation.
4. Use interviews to find out whether quality goals and roles improved understandability.

5. Use interviews and coding of user stories to find out how hard or easy it was to transform higher level goals to implementable user stories.
6. Use interviews to find out if creating goal models created overhead
7. Use literature review, interviews and models history from repository to verify if the modelling approach suits an iterative development.
8. Use the models history comparison to verify how much the requirements changed over time.
9. Use interviews to find out the effort taken to reflect changes back.
10. Use lines of code(LOC) from repositories and interviews to determine how fast will it go from idea to code.
11. Use interviews to get insight into the tooling part in the overall modelling activity.

Interviews require the most effort, but also give the most data to analyze.

3.3 Data sources

There are a several possible sources of data for a case study: interviews, surveys, focus groups, observations, metrics, archival data [28, 29].

Next data was collected during current case study project:

Interviews results

Interviews were held to answer research questions defined above in Chapter 3.2. As the project with an iterative requirements gathering is in progress, it is be best for the research if interviews would happen few times: after elaborating first branch of requirements, after every iteration or at the end of the project.

For this project only one set of interviews was conducted, interviews took place when planning session and three iterations already ended.

Goal models

Goal models were created during project's RE planning sessions. The models were saved with history and changes recordings times.

Source code

Source code created for a project under investigation is kept in version control system, which allows to observe lines of code (LOC), changes in LOC and time between LOC changes in case of need.

Meeting notes and recordings

Project meetings are video recorded and meeting notes are kept during the meetings and saved for future references.

According to the design of this case study, interviews are playing the biggest part in the research and give the most outcome. Other data sources are used mostly to confirm or deny statements received from the interviews.

3.4 Analysis procedure

Analysis procedures are applied to reduce and organize the data to provide a chain of evidence towards a conclusion. Different types of data are collected during this case study, but in the scope of this master thesis we are concentrating on interviews analysis, as it holds the answers to the most of the questions that interest us.

The second most important source of information are goal models, but goal models analysis [13] is a large topic and is out of scope for this thesis.

Analysis of interviews

The analysis of interviews is based on coding (Chapter 4.5) results. To code the interviews, a set of labels or codes is formulated based on research interests and assigned to phrases or sentences from the interviews. Results of the coding are analyzed per theme and presented. Each theme corresponds to a correct research design question from Chapter 3.2 or is introduced during interviews coding.

Codes are combined in tables, compared to each other and analyzed to evaluate aspects of the AAOM method from Chapter 3.2.

Analysis of interviews is explained in details in Chapter 5.

4 Data collection: interviews

4.1 Introduction

In this chapter, data collection topic is discussed as it is one of the main activities of a case study research. This activity is required to select data sources and organize the raw data received in a structured way to be able to analyze it afterwards.

It was already mentioned in Chapter 3.2 that properly conducted and analyzed interviews are providing most of the data to answer research design questions. That is the reason why this chapter is concentrating on the preparations of the interviews, interviews' conduction and preparation for analysis of the interview results.

Other data that needs to be collected (goal models, source code, meeting notes, meeting recordings) does not require any additional actions to be kept and thus is omitted.

4.2 Planning the interviews

According to [29], interviews are a first degree data collection technique that involves collecting data in real time involving direct contact with interviewees. To properly conduct the interviews, following items need to be defined: interview questions, interview session structure, interviewees roles and number.

People of different roles were chosen to be interviewed: three customers, one analyst and one developer. The analyst and developer was the same person, for whom one interview was conducted but different questions were asked considering either analyst or developer's point of view.

It was planned to have at least two interview sets: one in the middle of the project and one at the end. However, due to circumstances only one set(mid project) was conducted to collect the Lost&Found project's participants opinions about the AAOM method.

Interviews were semistructured: questions were planned, but order of the questions was not so important and could have been changed during the interview depending on a discussion flow and interviewee answers to preceding questions.

The interview timeglass model is used so that interview begins with more broad questions and is followed by more specific questions, at the end of the interview again broad questions are presented.

4.2.1 Interview questions

Sets of questions for interviews were prepared to answer the research design questions stated in Chapter 3.2. Different set of questions were defined for each role (client, analyst, developer) in a project. All prepared questions had no predefined answers. Questions were composed in such way that interviewees could not answer "yes" or "no", but had to express their own opinions.

The complete list of interview questions for client, analyst and developer is provided in Appendix A.

4.3 Interviews conduction

A pilot set of interviews was conducted at the moment when first plannings were done and then three sprints were finished. Four interviews were conducted overall: three for the clients and one combined interview for developer/analyst. Interviews were conducted by a researcher.

Each interview session was planned to take around one and a half hours and started with an introduction and then continued to the questions to evaluate different aspects of the AAOM method following the timeglass interview type mentioned earlier.

Interviews were audio recorded in MP4 files that were later used for post-interview activities and analysis.

4.4 Preparing the interviews for analysis

Before the interview analysis can be started, recorded interviews need to be transcribed to text files, which was done manually by a researcher. The transcripts received were reviewed by the interviewees as a final possibility for corrections and clarifications, but no changes to existing transcriptions were made. As a next step, a coding technique (described in Chapter 4.5) was used next to prepare transcripts of the interviews for the analysis.

4.5 Coding the interviews

4.5.1 What is the coding

In [12], 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, that could be organized by themes or categories. Each code represents a phrase, or few phrases from interviews.

Coding was made by using NVivo⁹ qualitative data analysis software trial version.

There are two types of codes possible:

A priori codes - predefined codes to check pre-existing theories, usually derived from questions.

Grounded codes - codes coming from the data, from the interviews in this case. Unexpected new findings might appear.

4.5.2 Codes

Since the objectives of the research are captured in the research design questions(Chapter 3.2), codes were formulated based on those questions.

Following themes were predefined before the coding(a priori themes):

⁹<http://www.qsrinternational.com/>

Table 2: Predefined coding themes

Theme name	Theme description
Benefits	To see what are the benefits of using the method from user perspective
Collaborative Modelling	Improving communication between the client and the development team. Having everyone on the same page
Elaboration Sessions	Sessions content, duration, suitability
Expectations	Check if users expectations were met
Method Clarification	Participants' understanding of the method and its details
Method Comparison	Comparing to other methods
Participation	Getting objective opinion about how the method includes everyone into the project
Time Taken for Modelling	How fast can we move on with this method
Tools Usage	Covers tooling importance for the method, how much difference the tool makes (clearness, confusion)
Visual Representation	Understanding if visual approach is suitable for RE

New grounded themes were introduced during interviews coding. Next topics or themes united those codes:

Table 3: Grounded coding themes

Theme name	Theme description
Drawbacks	Negative moments about the methods mentioned in interviews
Modelling Suitability	Finding out if modelling is suitable for different kinds of projects
New Ideas	Proposals for method improvements or project setup improvement

The complete list of the codes inside those themes are presented in the following Chapter 5 and are also enumerated in Appendix B.

Interviews' data analysis is done now by using the list of codes from interviews' codings.

5 Interviews analysis

5.1 Introduction

Once the interviews are done and coded, the analysis commences. The main goal of the analysis is understanding whether theories about the AAOM method are valid: collected data is used to find answers to questions specified in Chapter 3.2. Any non-expected data found during the interviews is also taken into consideration. In this chapter, codes of interviews are prepared for analysis, every theme of codes is analyzed separately, and some conclusions are made based on the analysis results.

5.2 Interviews' codes analysis

The analysis of interviews' codes is performed and some adjustments are applied to facilitate the analysis of interviews' codings: attributes are added to every code, and a formula is introduced to understand code's relative values.

5.2.1 Codes attributes

Two additional attributes are introduced to ease codes analysis: Polarity and Type. The Polarity is used to determine an emotionality of a code, and the Type is used to find if it is a recommendation from the user or a remark about something.

Polarity

Possible values: Negative, Positive, Neutral.

Code polarity shows how code was presented by the interviewees - whether negatively, positively or neutrally.

Positive polarity is for positive opinion.

Neutral - fact or statement without a positive or negative emotion.

Negative - pessimistic opinion.

Type

Possible values: Statement, Suggestion.

Type attribute is added to indicate if code is a Statement or a Suggestion.

Statement describes the current situation.

Suggestion is a proposed change.

Polarity + Type

A combination of Polarity and Type is used to describe whether code proves, contradicts or has no relation to research.

Negative suggestion - indicates a need to change existing part of the process.

Positive suggestion - is an additional feature proposed by the customer.

Neutral suggestion - came out to be a suggestion not related to the AAOM method itself.

Negative statement - something is not working in the method, but there is nothing to do about it.

Positive statement - something is considered to be working or suitable.

Neutral statement - opinion or a statement of affairs not related to the research question.

5.2.2 Formula - finding a value of the code

We developed a simple formula to get an idea about what codes have more value in terms of mentions. Following parameters have an influence on code value and are included in the formula:

References - how many times the code was mentioned in interviews. More references increase the value.

Sources - how many different people mentioned one code. More sources increase the value.

Role experience - what was the interviewee's experience in his role. Every reference from experienced role adds more value to the code.

All these metrics define the value of the code. The formula below was created to sort codes based on mentioned metrics:

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

The results of the formula applied to the list of codes can be found in appendix B.

5.3 Analysis of interviews codes by theme

Codes for every theme are analyzed below. Codes in each theme are ordered by its value obtained by using the formula - most valuable codes upfront.

The table of codes for each theme is provided with codes names, polarity, type and formula value. Codes are sorted by the formula value descendingly so that the most important codes are brought out topmost.

5.3.1 Benefits

Codes for this predefined theme are listed in Table 4.

Table 4: Codes for theme "Benefits"

Code	Polarity	Type	Value
Secure feeling for project direction	Positive	Statement	10
Mutual communication	Positive	Statement	7
Discover new angles	Positive	Statement	4
Intuitively understandable	Positive	Statement	2
Easily modifiable	Positive	Statement	2
Constructive modelling	Positive	Statement	1
Estimate work ahead	Positive	Statement	1

All the codes in this theme are positive statements and support the theory that the AAOM method helps to understand requirements for the project, keeps everyone on the same page and most of all - clients feel secure about the project direction.

5.3.2 Collaborative Modelling

This theme is derived from the method itself, improving communication between client and the development team. Table 5 lists the codes for "Collaboration Modelling" theme.

Table 5: Codes for theme "Collaborative Modelling"

Code	Polarity	Type	Value
Having everyone on the same page	Positive	Statement	18
Improved understandability	Positive	Statement	14
Pinpointing problems	Positive	Statement	13
Involving participants	Positive	Statement	5
Composing goal models should be more structured	Neutral	Suggestion	2
Few feelings about collaboration	Neutral	Statement	1
Sharing tasks well	Positive	Statement	1

Given codes mostly support theory that the AAOM method is good for collaboration and it helps to involve everyone in the same room to work together.

Though one suggestion made by analyst was that composing of goal models could be more structured.

5.3.3 Drawbacks

It is a grounded theme, came from interviews, sometimes is related to surrounding environment, not the method itself. Table 6 lists the codes for theme "Drawbacks".

Table 6: Codes for theme "Drawbacks"

Code	Polarity	Type	Value
Experienced participants required for full potential	Negative	Statement	22
Better guide for analyst needed	Negative	Suggestion	14
Analyst has the most responsibility	Neutral	Statement	5
Method might be overhead for smaller projects	Neutral	Statement	5
Hard to move from goals to user stories	Negative	Statement	3
Goal models too general, need more technical details	Negative	Statement	2
Initial user stories take time	Neutral	Statement	2
Initial models need refinement	Neutral	Suggestion	1
Starting from scratch should be more structured	Negative	Suggestion	1

As the underlining theme name suggests, most of the codes here have a negative meaning.

The most mentioned code states that experienced participants are needed to get the most of the AAOM method, many other codes suggest that better guidance and trainings on how to use the method are needed: maybe examples, templates on how to go from an idea to user stories in one project, better guide for analyst.

The code "Goal models too general, need more technical details" is arguable: it might depend on a team and participants, but by general rule, user stories should be technical, not goal models.

5.3.4 Elaboration Sessions

Predefined theme. Sessions content, duration and suitability is considered. Table 7 lists the codes for theme "Elaboration Sessions".

Table 7: Codes for theme "Elaboration Sessions"

Code	Polarity	Type	Value
Session length suitable	Positive	Statement	5
Sessions could be earlier, less tired	Positive	Statement	2
Shorter sessions for inexperienced in analysis	Negative	Suggestion	2
New ideas since previous meeting offloading	Positive	Statement	1
One topic per meeting	Positive	Statement	1
Only new info on sessions	Positive	Statement	1

Most of the codes here are positive statements confirming that session length and content were suitable. Meetings length and time of a day is important and should be carefully selected, too long meetings should not be forced.

It is also mentioned that people have different attention spans, and for not experienced in analysis 1,5 hour long sessions(used in this concrete project) might be too long.

5.3.5 Expectations

Predefined theme for checking if user expectations were met. Table 8 lists the codes for theme "Expectations".

Table 8: Codes for theme "Expectations"

Code	Polarity	Type	Value
Updates to models and user stories	Neutral	Statement	8
Working results	Neutral	Statement	5
Extendable implementation	Positive	Statement	4
Need more resources to accomplish goals	Neutral	Suggestion	1

Considering emotionality of the given codes it might be said that expectations are met, but not exceeded: work is in progress, some work is already done, goal models are expected to be updated and grow.

Client wants more people to work on the product, but this code is related to the project, not the method itself.

5.3.6 Method Clarification

Predefined theme. Participants understanding of the method and its details. Table 9 lists the codes for theme "Method Clarification".

Table 9: Codes for theme "Method Clarification"

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

Negative statements from this theme are of interest to us. It turns out that not all actions of the process are completely clear to all the participants. First of all, the way of finding the lowest level of goal models and getting to user stories from there is not explicit to half of the participants. Secondly, another ambiguous concept is a quality goal and a method of its transformation to user stories. It seems that these concepts need to be clarified better in the description of the AAOM method.

5.3.7 Method Comparison

Predefined theme, derived from research question. Table 10 lists the codes for theme "Method Comparison".

Table 10: Codes for theme "Method Comparison"

Code	Polarity	Type	Value
Making notes	Positive	Statement	1
Modeling in Scrum	Positive	Statement	1

Initial idea was to compare the AAOM method to other methods, but there was not much to compare as participants were not very experienced.

As the result, comparison with other methods is not sufficient and did not provide enough

results. Though in addition with other test cases data those results might still be useful.

In one code interviewee compares the AAOM method to making notes, though usage depends on the result one wants to achieve. Another code mentions that extra work is done compared to Scrum but the method also gives useful information in a visual view, which is a bonus.

5.3.8 Modelling Suitability

Grounded theme, was introduced during coding of the interview and is used for finding out if modeling is suitable for this specific project. Table 11 lists the codes for theme "Modelling Suitability".

Table 11: Codes for theme "Modelling Suitability"

Code	Polarity	Type	Value
Clarifies what needs to be done	Positive	Statement	10
Organizing thoughts	Positive	Statement	6
Modelling suits into various project setups	Positive	Statement	5
Quality goals more for analyst and developer not client	Neutral	Statement	1

Again given codes support the theory about the method helping to focus on figuring out objectives and organizing thoughts to express client's feelings.

Method might be combined with other systems and tools, might suit to various projects setups with some limitations. There is also an opposite code "method might not suit small projects".

Interesting feedback is given about clients not being interested in quality goals.

5.3.9 New Ideas

Grounded theme, was introduced during coding of the interview and consists of ideas that interviewees mentioned. Table 12 lists the codes for theme "New Ideas".

Table 12: Codes for theme "New Ideas"

Code	Polarity	Type	Value
Link metrics to goals	Neutral	Suggestion	2
Models can be used for system documentation	Positive	Suggestion	2
Quality goals holding technical details	Positive	Suggestion	2

One of the client was interested in assigning financial values to the goal and comparing if it is needed. This could be achieved.

Another idea was to use goal models as a system documentation. Also could be done: the AAOM method is not preventing users from creating other models besides goal models that would elaborate in more details of how the system works.

Third suggestion was that quality goals could hold also technical information and, again, this could be done easily with the AAOM method.

5.3.10 Participation

Predefined theme that tries to get objective opinion about how the method includes everyone into the project. The codes for this theme are listed in Table 13.

Table 13: Codes for theme "Participation"

Code	Polarity	Type	Value
Participation level is satisfactory	Positive	Statement	13
Specific stuff didn't felt close to heart	Negative	Statement	4
Background with ICT helps to participate	Neutral	Statement	2

In this theme we tried to find out if participation level matches the expectation and it seemed to be so.

It was one very interesting point that clients lost interest when getting deeper into details.

5.3.11 Time Taken for Modelling Activities

This theme is predefined to measure how fast is it possible to move on with this method. The codes for this theme are listed in Table 14. Here, the unit of measurement happened to be not the time, which might vary a lot depending on the project, but participant's subjective feelings about the time spent on method activities.

Table 14: Codes for theme "Time Taken for Modelling Activities"

Code	Polarity	Type	Value
Quickly to development	Positive	Statement	11
Refining goal models fast	Positive	Statement	5
Previous method used before AAOM took lot of time	Positive	Statement	5
Time used effectively	Positive	Statement	4
From idea to user story just enough time	Neutral	Statement	1

All the codes support the theory that the AAOM method accelerates refining the requirements. The goal models definition process itself went fast. Clients are also very happy that development started quite fast and satisfied with time effectiveness.

5.3.12 Tools Usage

This a priori theme covers tooling importance for the method. The question to answer is how much difference the tools make. The codes for this theme are listed in Table 15.

Table 15: Codes for theme "Tools Usage"

Code	Polarity	Type	Value
Manual integration worked but a lot of extra work	Negative	Statement	9
Good enough for starters	Neutral	Statement	6
Need more integration	Negative	Suggestion	5
Commercial better	Negative	Statement	4
Commercial expensive	Negative	Statement	4
Dedicated tool support	Neutral	Statement	2
Easy and flexible	Positive	Statement	2
Interesting to use new tools	Positive	Statement	1
New tools need training	Negative	Statement	1

All of the tools used for this project were free tools. As codes suggest, chosen set of tools was good enough and it was possible to do the work by using them. Tools were not connected together and manual integration was required.

It was mentioned by several codes that commercial tools could have been better, if there would have been money for them in a project budget.

5.3.13 Visual Representation

Predefined theme to understand if visual approach is good for requirement engineering. The codes for theme "Visual Representation" are listed in Table 16.

Table 16: Codes for theme "Visual Representation"

Code	Polarity	Type	Value
Goal model representation - benefit	Positive	Statement	25

The codes of the interviews confirm that using visual approach for requirements engineering is natural, intuitive and easy to understand.

5.4 Analysis summary

A lot of information was gathered during interview sessions. A coding technique (Chapter 4.5) was employed to organize and analyze the data. Different aspects of the AAOM method were evaluated while taking into consideration limitations of a given case study.

5.4.1 Gathered data limitations

We admit and accept that results gathered during the case study from real-life project have a number of limitations and depend on concrete project's setup. In this project three people out of four participants played a client role. Thus, collected information is mostly based on the client-side opinion about the AAOM method.

Only two out of four people had experience participating in software development process and thus, we get a lot of feedback from inexperienced participant's point of view. Opinion from inexperienced people is also important since this shows how easy it is to adopt AAOM. In real life, it is often the case that a client is quite unfamiliar with the technical state-of-the-art and the ideas explained to developers are often hard to grasp and error-prone.

The limitations we have in this case study is a gap to be fulfilled in future work on the evaluation of the AAOM method and is mentioned in Chapter 6.3.

5.4.2 Results

Different aspects of the AAOM method were evaluated and collected positive feedback is organized into the following categories:

Visual approach

The visual approach of the AAOM method was found to be natural, intuitive and easy to understand.

Collaboration and participation

The AAOM method stimulated everyone's collaboration and commitment to common work.

Extendability

Participants have different ideas about how to supplement method with additional features, and all of the ideas are implementable. Thus, we can claim

that the method is easily extendable.

Requirements elicitation

Helps to understand requirements for the project, keeps everyone on the same page and most of all - clients feel secure about the project direction. Modelling helps to guide thoughts of the client and transform them into implementable user stories.

Time and speed

The AAOM method accelerates the requirements specification and goal models definition process went fast even for a multi-goal project. The development started quickly. Creation of goal models during requirements elicitation during the planning session does create a bit of an overhead, though it was still fast, but it resulted in a visual model that helped a lot to understand the goals and directions of a system to be built.

Negative reaction appeared regarding following topics:

Quality goals

It turned out that even after the main planning phase was over, not all participants do understand how to connect quality goals to user stories and how to use quality goals at all.

Process of transforming goals to user stories

Some participants notice that it is not fully understandable how to move from goal models to user stories and how to get to know if the lowest level of goal models is reached.

The following categories depend on a particular team or project setup:

Tools

A tooling part came out to be important. Free tools used were good enough, but commercial tools would have been better. The tool choice depends a lot on a project and on the list of tools that are already in use in a development team.

Clients lost interest when getting deeper into details

It is mentioned by several clients that getting into the technical details of implementation of user stories is not interesting to them. This might depend a lot on the customer's background, but the method itself does not specify if

user stories implementation discussions should happen in front of clients, or inside the development team. It is up to the project's team to decide.

Elaboration sessions

Elaboration sessions length and time of a day are important. Shorter meetings are recommended instead of longer ones, but the team decides what suits them.

Overall, the modelling approach seems to suit iterative agile development.

5.4.3 Method improvements

Based on negative feedback, several method improvements could be carried out:

- Better practical guides and materials on how to conduct activities based on AAOM method are required.
- Examples from real life projects could be included in training materials.
- Transforming goal models to user stories proves to be difficult and needs more experimenting and guidance.
- The quality goals role in the method should be explained better.

6 Conclusion

6.1 Summary

In this thesis, we evaluate the AAOM method as a novel visual approach to an RE for smaller projects in agile software development. A case study research approach was used to conduct such an evaluation. The method was applied in a real-life settings for developing the Lost&Found mobile app from scratch.

Different sources of evidence were used to collect the data to answer defined research questions. Interviews of project's participants provided the most important input for method's assessment.

An analysis of interviews was performed to check different aspects of the method. Interviews were coded by looking for predefined concepts and categories in the data for analysis units. Received codes were combined into tables and analyzed.

6.2 Answers to research questions

The research questions defined in introduction have been answered:

RQ1: How to evaluate AAOM with suitable research method?

The AAOM method is evaluated with the use of a case study research approach by analyzing the method from different angles: visual approach, time used for modelling activities, participation level, understandability of result models, suitability for iterative development and others. The Lost&Found mobile app prototype has been developed as a demonstration case for a case-study research. The AAOM method was used during planning phase for requirements elicitation for this project.

RQ2: How to collect data for an AAOM-evaluation?

Most of the data for AAOM-evaluation is collected during interviews of participants, and the thesis is concentrating mostly on interview preparation, conduction and analysis. Proper questions should be asked to receive meaningful answers. Meeting notes and recordings, goal models, and app source code are also collected, but are out of scope of this master thesis.

RQ3: How to analyze the collected data?

In scope of this master thesis we are concentrating on interviews analysis, as it gives the most information on evaluation of the AAOM method. Coding of interviews is performed by themes to assess different aspects of the AAOM method. Collected codes are organized into tables and then analyzed in Chapter 5.

RQ: How suitable is AAOM for RE in agile software development?

The evaluation of a case study of the Lost&Found project proves the applicability of AAOM for RE in an agile software development process. 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.

6.3 Limitations, open issues, future work

Further studies and investigation will continue going on with a purpose to investigate the AAOM method's utility for requirements engineering in an agile software development projects. This master thesis is part of a multi-case study for evaluation of the AAOM's method. The AAOM method will be employed and analyzed in other projects, and overall evaluation of all case studies will be performed.

During analysis of the AAOM method in current master thesis, the following directions of future research were determined:

Improvements to the AAOM method

Several method improvements could be implemented based on a feedback from project's participants:

1. Role of quality goals should be explained better.
2. The transformation of goal models to user stories proves to be difficult and needs more experimenting and guidance.
3. Better guides and materials could be added on how to conduct the method's activities.

More feedback from analysts

The analyst plays an important role when using this method, so more feedback from analysts could be gathered in the future.

Compare AAOM to other RE methods

Opinions of experienced with agile and other RE methods participants are needed to be gathered to properly compare AAOM to other RE methods. This comparison would also help to understand how well the AAOM method could be combined with other agile methods.

Kokkuvõte

Antud magistritöös uuritakse ja hinnatakse Agiilse Agent-Orienteeritud Modelleerimise (AAOM) meetodit. AAOM on meetod kiireks nõuete kogumiseks, mis kasutab kahte põhikomponenti. Esimeseks on eesmärgimudelid agent-orienteeritud modelleerimise raamistikust süsteemi eesmärkidest ülevaate andmiseks. Teiseks põhielemendiks on kasutuslood (user stories), mis on pärit agiilsetest arendusmeetoditest. Kombineerides eesmärgipuud kasutuslugudega saame meetodi, mille abil annab teostada nõudmiste kogumist.

Antud uurimistöö põhieesmärk on leida, kui kasutuskõlblik on AAOM agiilises tarkvara arenduses. Et vastata sellele küsimusele, on vaja kõigepealt selgitada välja, mis meetodiga hinnata meetodi kasutuskõlblikkust, sellest lähtuvalt panna paika kuidas ja mis andmed korjata hindamiseks, ning lõpuks kuidas korjatud andmeid töödelda ja analüüsida.

Valitud hindamismeetodiks sai juhtumiuuringul põhinev meetod. Juhtumiuuringu käigus jälgisime kui lihtne või raske on AAOM-i kasutada reaalses arendusprojekti. Vaadeldavaks juhtumiks oli mobiilirakenduse Lost&Found arendamine, mille näol oli tegemist ka agiilse arendusmeetodiga juhitava projektiga. Projektil olid keerukad nõudmised, mis vajasid põhjalikku viimistlust ning see sobiski hästi AAOM-i kasutuse hindamiseks.

Projekti käigus koguti erinevaid andmeid: koosolekute märkmed ja salvestused, eesmärgimudelid, kasutuslood, muudatuste ajalugu ning rakenduse lähtekood. Kuid kõige rohkem väärtuslikku informatsiooni antud magistritöö eesmärgi saavutamiseks pakkusid projektis osalejate intervjuud. Seega, juhtumiuuringu andmehulgast analüüsitakse käesolevas magistritöös intervjuusid. Intervjuude ülesehitus valiti vastavalt teaduslikele meetoditele ja küsimused seadistati vastavalt uurimiseesmärkidele.

Intervjuude analüüs viidi läbi nende kodeerimise (coding) abil, mis toimus samuti vastavalt teaduslikele meetoditele. Saadud koodide abil õnnestus hinnata AAOM meetodi erinevaid aspekte: olemite visuaalne esitus, modelleerimise peale kulunud aeg, osalejate kaasamine, kokkusobimine agiilse arendusega, saavutatud mudelitest arusaamine jne. Kogutud koodid analüüsiti käesoleva magistritöö raames, et saada kinnitust uurimisküsimustele.

Juhtumiuuringu käigus analüüsitud Lost&Found juhtumi abil sai leitud, et AAOM meetod on kasutuskõlblik nõudmiste selgitamiseks agiilsete meetoditega juhitud arendusprojekti. Analüüs näitas, et meetodi visuaalne esitus aitab nii klientidel kui ka arendusmeeskonnal mõista süsteemi suuremaid eesmarke, meetod on kergesti laiendatav, julgustab kostööd ja osavõttu ning ei võta palju aega.

A Appendix - Interview Questions

A.1 Client questions

1. Are the main concepts of agile AAOM clear? Can you explain in a couple of sentences how you understood them?
2. Is the visual representation of your intentions in the form of goal models understandable?
3. Did collaborative modelling involved you more into the process?
4. How do you evaluate your participation in goal models creation? Wanted to do more or less? More high or low level?
5. How do you evaluate your participation in goal models creation? Wanted to do more or less? More high or low level?
6. How did elaboration-session execution work for you? Time it took, pauses taken, tooling setup, suitable time?
7. How clear was the process from your general idea to user stories (lowest leaf)?
8. Did you understood user stories presented to you?
9. Did quality goals and roles attached to functional goals appear reasonable and provide extra value?
10. Did the linking of roles and user stories to quality goals seem logical and simple?
11. How satisfied were you with how much time it took to get from the main idea to user stories?
12. Did tools used for modelling help to understand or create confusion?
13. How did the modelling method fit into the remaining activities needed to run the project?
14. What do you expect from future iterations?
15. Any additional remarks/questions?

A.2 Analyst questions

1. Are the main concepts of agile AAOM clear? Can you explain in a couple of sentences how you understood them?
2. Did goal models help to extract information from a client in an easy, structured and logical way?
3. Did collaborative modelling help to involve clients more into the requirements elicitation process?
4. Did quality goals give a valuable insight to an analyst?
5. Was the user story concept understandable as a piece of implementable value?
6. Was user story a sufficient and clear goal to achieve?
7. Did roles help to compose user stories?
8. How hard was it to move from general ideas to specific user stories?
9. How much time did it take to get goal models ready?
10. Did goal models help to ease introducing developers to the project?
11. How did tooling help/distract your effort to document models?
12. How did the requirements-elicitation sessions go? Did you feel a need for instructions about how to conduct an elicitation session?
13. How did the modelling method fit into the rest of activities needed to run the project?
14. What do you expect from future iterations?
15. Any additional remarks/questions?

A.3 Developer questions

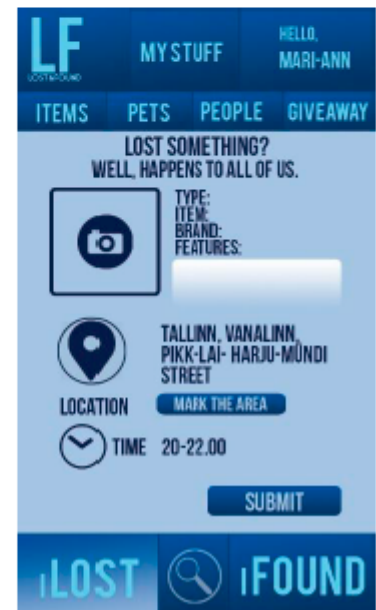
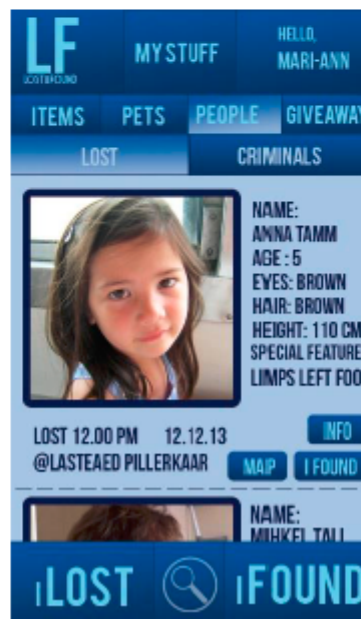
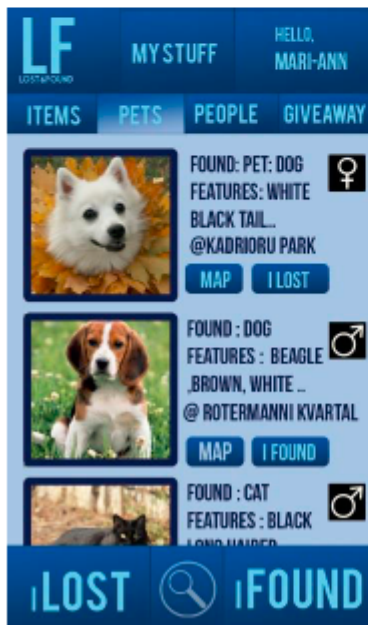
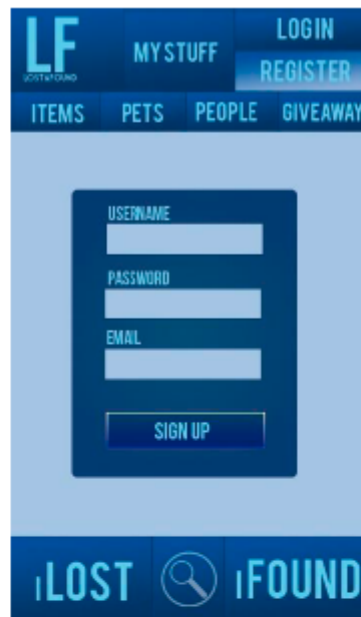
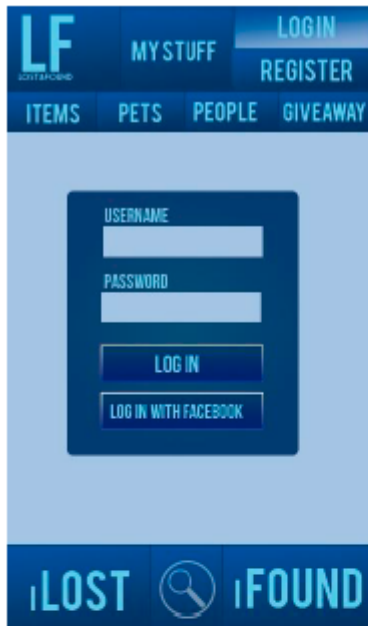
1. Are the main concepts of agile AAOM clear? Can you explain in a couple of sentences how you understood them?
2. Did you get the idea why and what you are about to implement?
3. Did you see what value you are about to deliver?
4. Was the work presented to you small enough for implementation? By definition of user story - small enough to be implemented in a day or two?
5. Did you notice how quality goals affect your tasks?
6. How much time did it take to implement the user stories?
7. How did the modelling method fit into the rest of activities needed to run the project?
8. What do you expect from future iterations?
9. Any additional remarks/questions?

B Appendix - Codes and Formula

Name	References	Sources	Experienced	Formula
Method clarification	48	5	15	255
Sequence of activities clear	7	4	2	30
User Story concept understandable	6	3	2	20
Goal Model understandable	6	4	2	26
Usage of Quality Goals understandable	5	4	1	21
From Goals to User Stories unclear	5	3	1	16
From Goals to User Stories logical	3	3	1	10
Usage of roles clear	3	3	1	10
User Story concept unclear	3	2	1	7
Quality Goals link to User stories unclear	3	2	3	9
Development process unclear	2	1	0	2
Goal model lowest level finding unclear	1	1	0	1
Roles useful for User Story creation	1	1	0	1
User Stories created by analyst unclear	1	1	1	2
Quality goals should have more details	1	1	0	1
Usage of Quality Goals unclear	1	1	0	1
Drawbacks	21	4	10	94
Experienced participants required for full potential	6	3	4	22
Better guide for analyst needed	4	3	2	14
Hard to move from goals to User Stories	3	1	0	3
Analyst has the most responsibility	2	2	1	5
Method might be overhead for smaller and really concrete projects	2	2	1	5
Initial models need refinement	1	1	0	1
Starting from scratch should be more structured	1	1	0	1
Goal models too general, more technical details needed	1	1	1	2
Initial User Stories take time	1	1	1	2
Collaborative Modelling	19	5	7	102
Having everyone on the same page	5	3	3	18
Improving understandability	4	3	2	14
Pinpointing problems	4	3	1	13
Involving participants	2	2	1	5
Composing goal models should be more structured	2	1	0	2
Few feelings about collaboration	1	1	0	1
Sharing tasks well	1	1	0	1
Tools Usage	17	4	2	70
Manual Integration worked but lot of extra work	3	3	0	9
Good enough for starters	3	2	0	6
Easy and flexible	2	1	0	2
Commercial expensive	2	2	0	4
Need more integration	2	2	1	5
Commercial better	2	2	0	4
Interesting to use new tools	1	1	0	1
Dedicated Tool support	1	1	1	2
New tools need training	1	1	0	1
Benefits	12	3	6	42
Mutual Communication	3	2	1	7
Secure feeling for project direction	3	3	1	10
Discover new angles	2	1	2	4
Intuitively understandable	1	1	1	2
Easily modifiable	1	1	1	2
Estimate work ahead	1	1	0	1
Constructive modelling	1	1	0	1

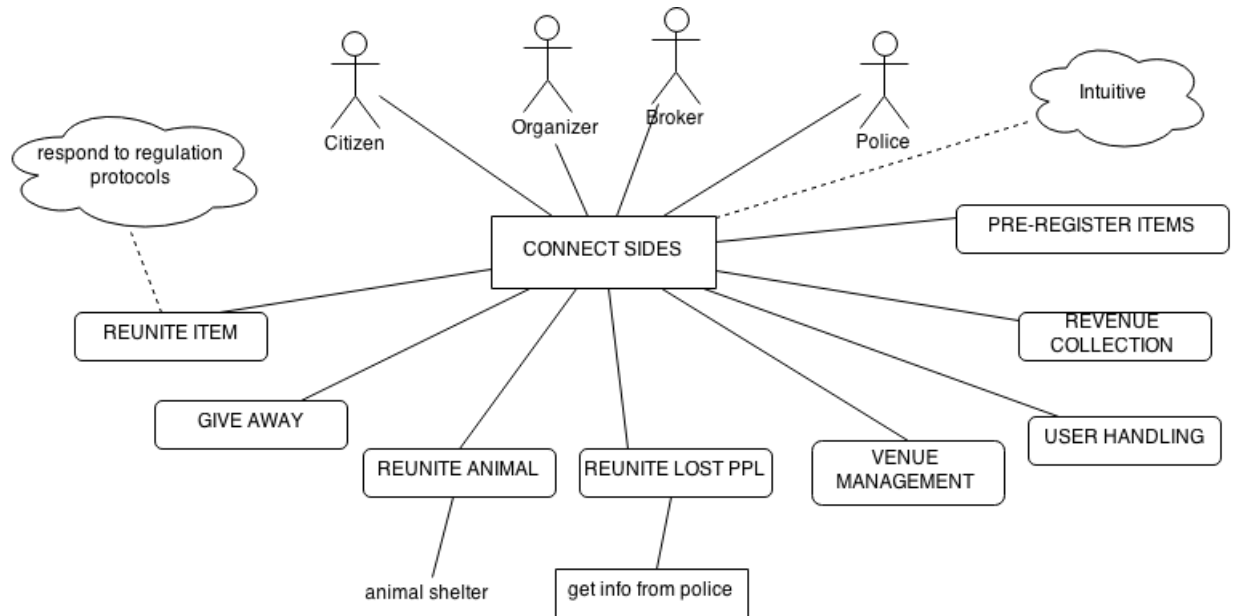
Time taken for modelling activities	11	4	5	49
Quickly to development	4	2	3	11
Before using method (agile AOM) took lot of time	2	2	1	5
Refining goal models fast	2	2	1	5
Time used effectively	2	2	0	4
From idea to User Story just enough time	1	1	0	1
Modelling suitability	9	4	2	38
Organizing thoughts	3	2	0	6
Clarifies what needs to be done	3	3	1	10
Modelling suits into various project setups	2	2	1	5
Quality goals more for analyst and developer not client	1	1	0	1
Expectations	8	3	5	29
Update to models and User Stories	3	2	2	8
Extendible implementation	2	1	2	4
Working results	2	2	1	5
Need more resources to accomplish goals	1	1	0	1
Participation	7	3	2	23
Participation level satisfactory	4	3	1	13
Specific stuff didn't felt close to heart	2	2	0	4
Background with ICT helps to participate	1	1	1	2
Elaboration sessions	7	3	3	24
Session length suitable	2	2	1	5
One topic per meeting	1	1	0	1
Sessions could be earlier, less tired	1	1	1	2
Only new info on sessions	1	1	0	1
Shorter sessions for inexperienced in analysis	1	1	1	2
New ideas since previous meeting offloading	1	1	0	1
Visual representation	6	4	1	25
Goal model representation - benefit	6	4	1	25
New Ideas	3	2	3	9
Quality Goals could hold technical details	1	1	1	2
Models can be used for system documentation	1	1	1	2
Link metrics to goals	1	1	1	2
Method Comparison	2	2	0	4
Modeling in Scrum	1	1	0	1
Making Notes	1	1	0	1

C Appendix - Design concept for Lost&Found mobile app

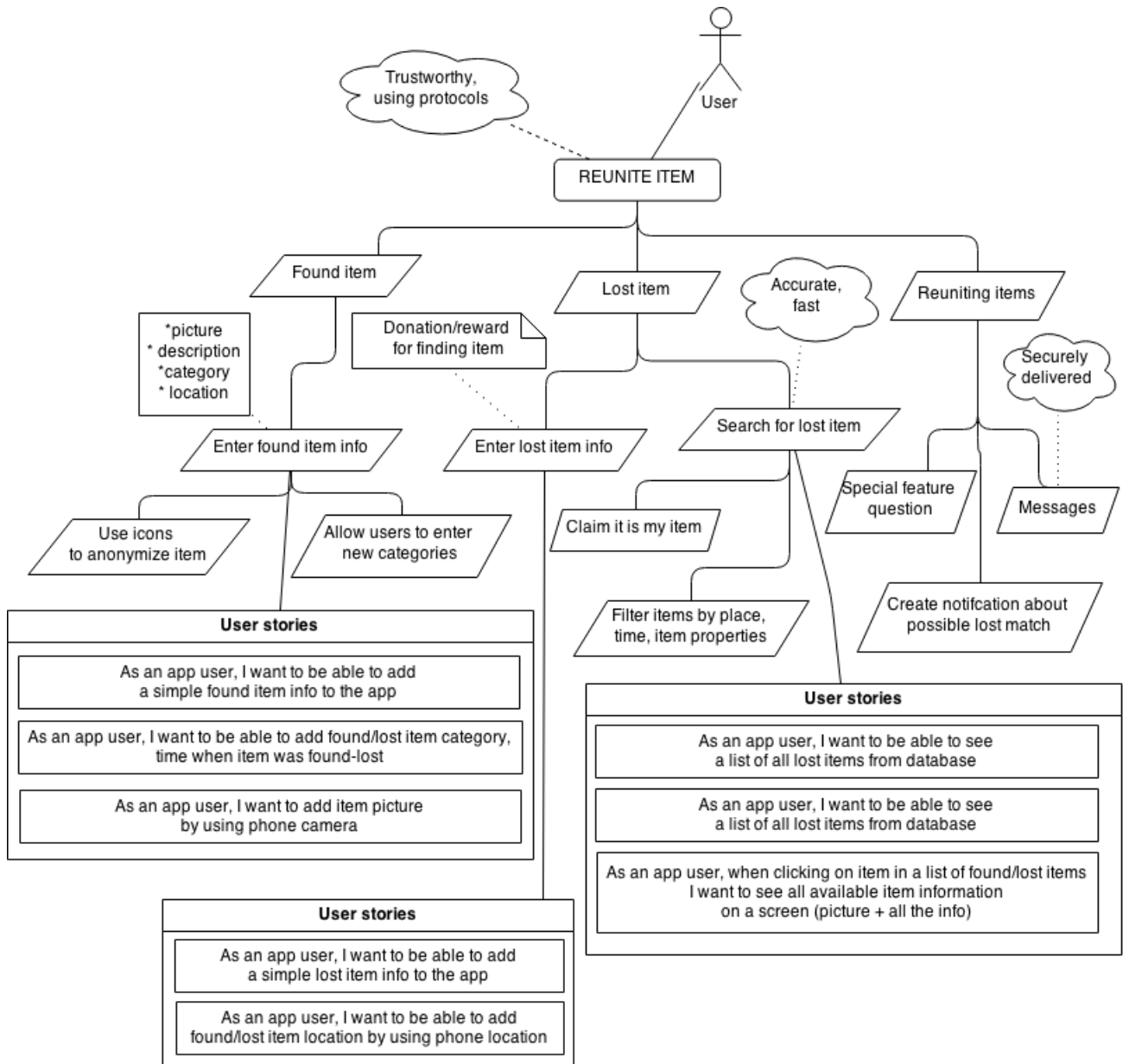


D Appendix - Goal models for Lost&Found mobile app

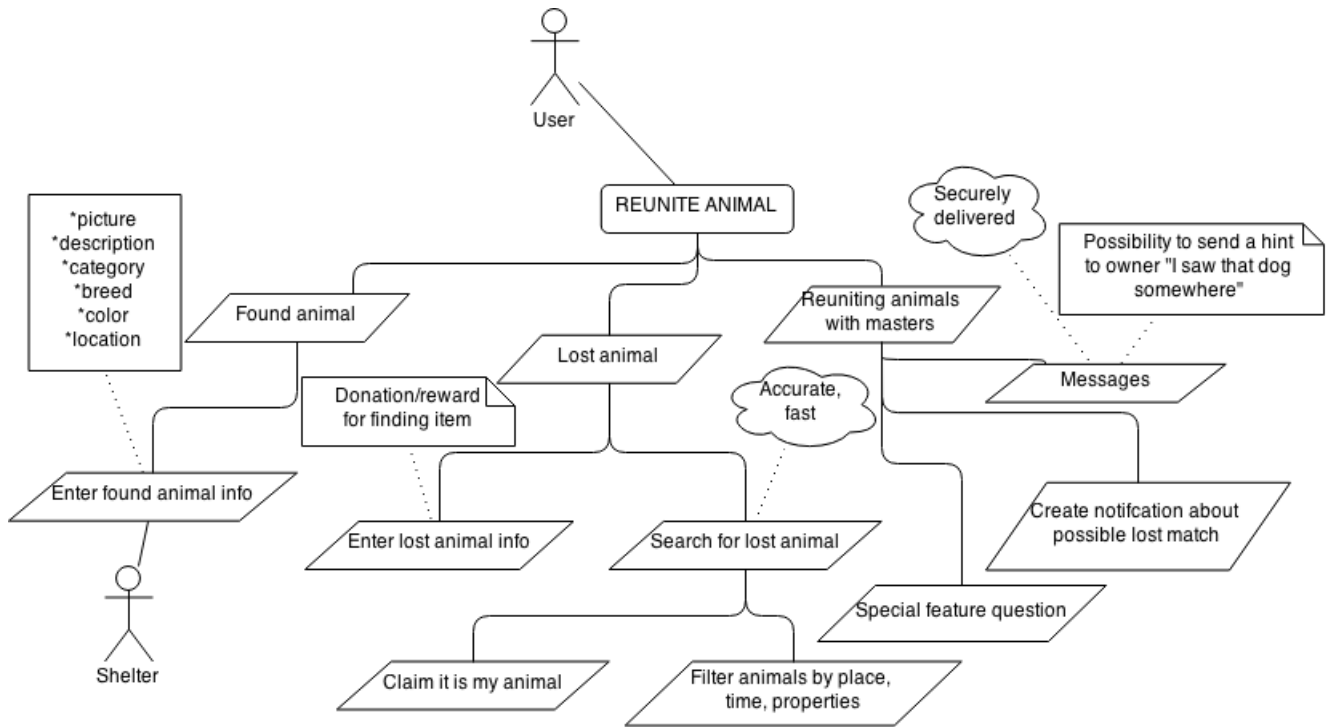
D.1 Main goal model



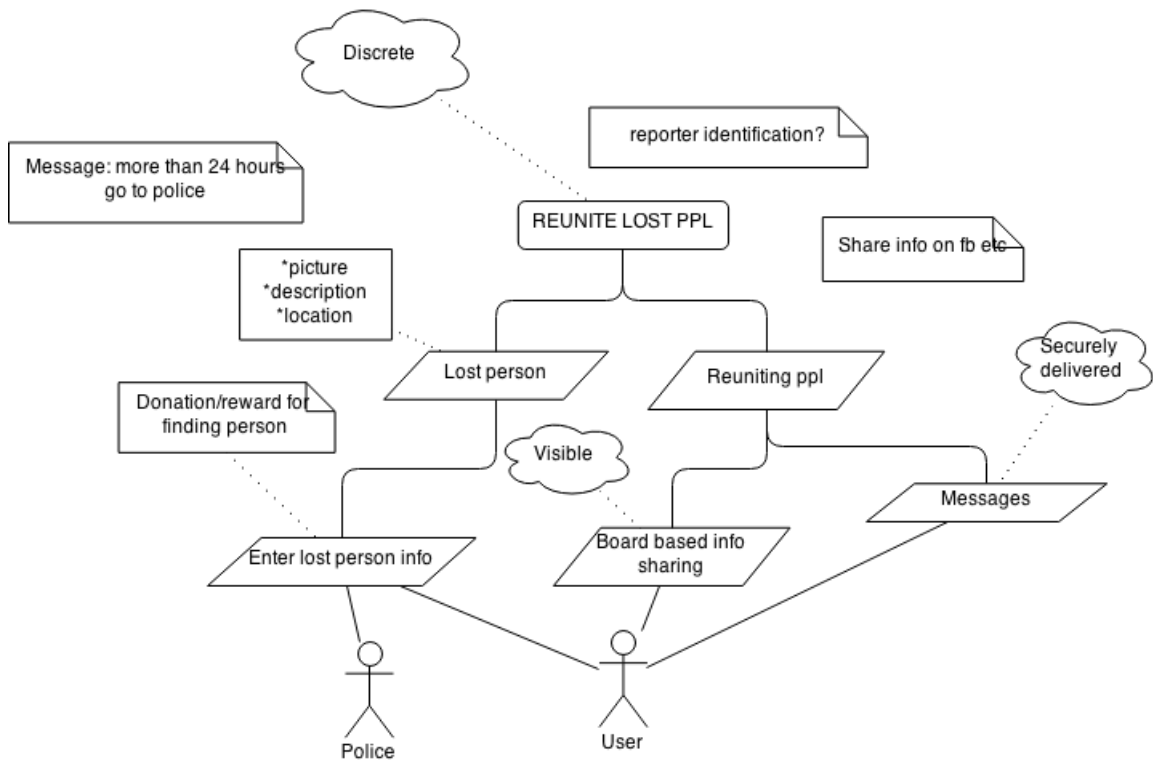
D.2 Goal model for lost items



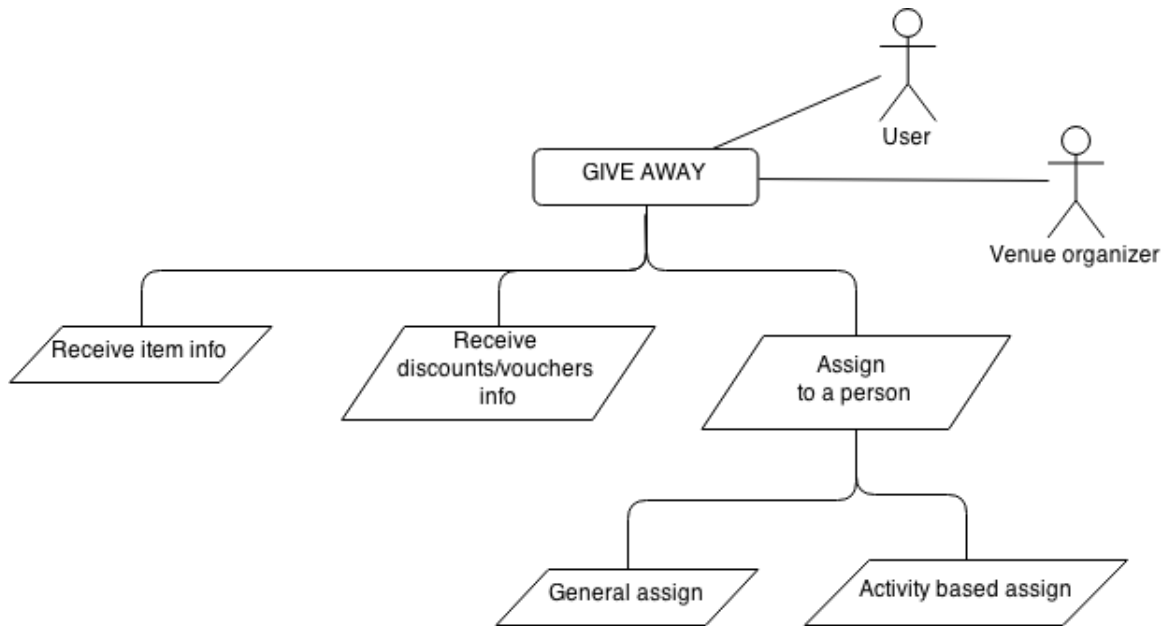
D.3 Goal model for lost animals



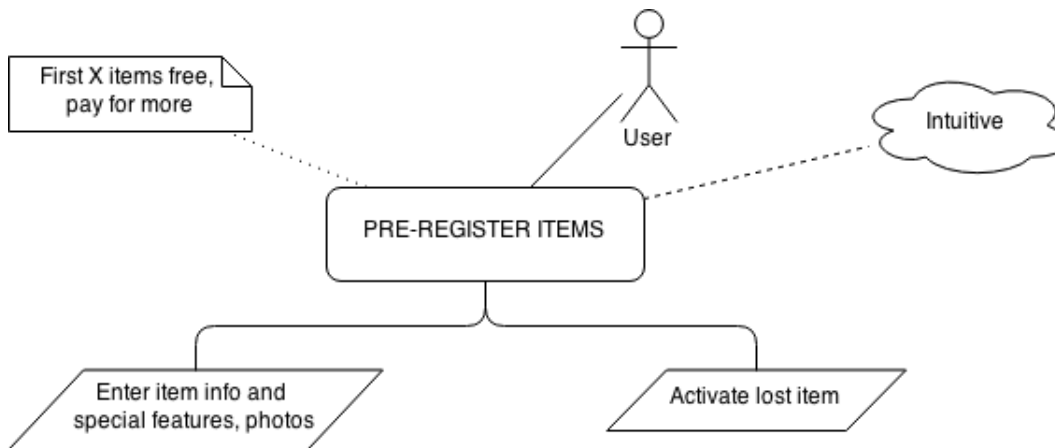
D.4 Goal model for lost people



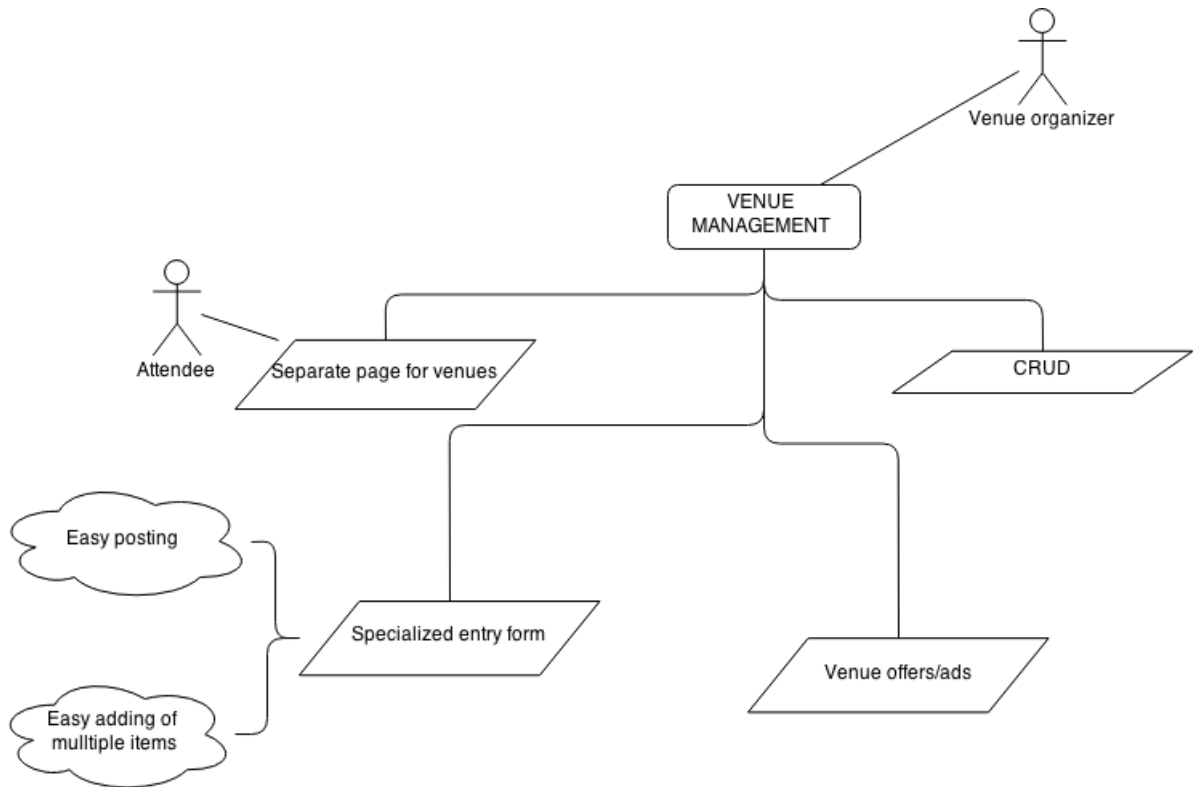
D.5 Goal model for giveaways



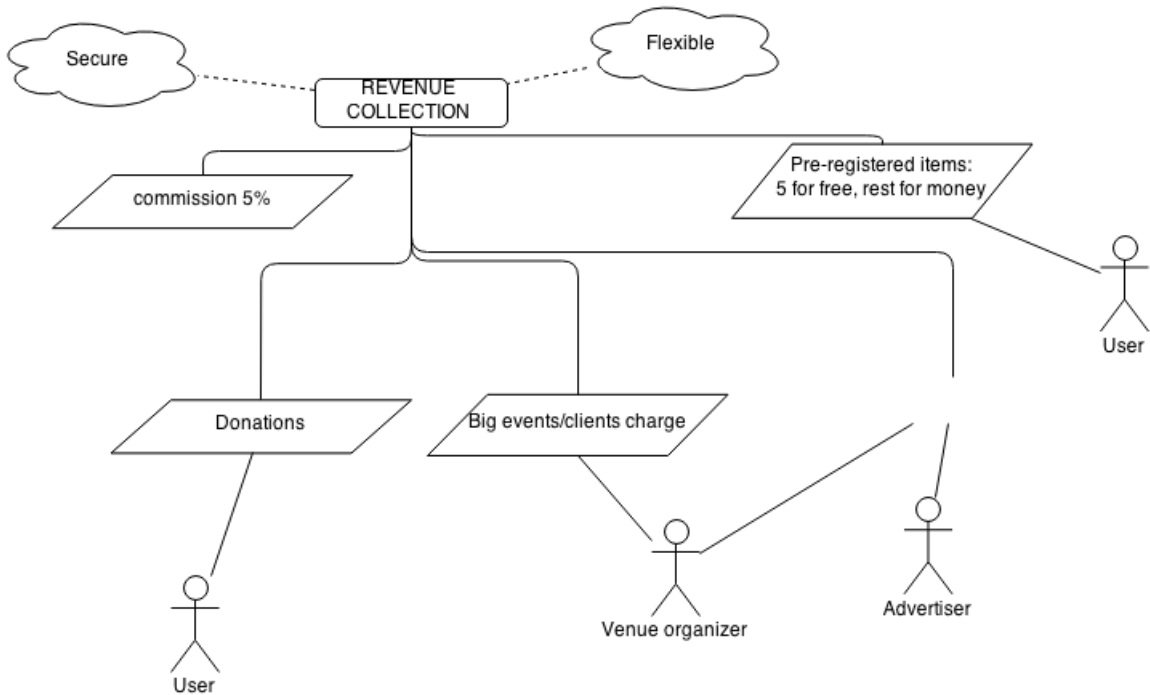
D.6 Goal model for pre-registered items



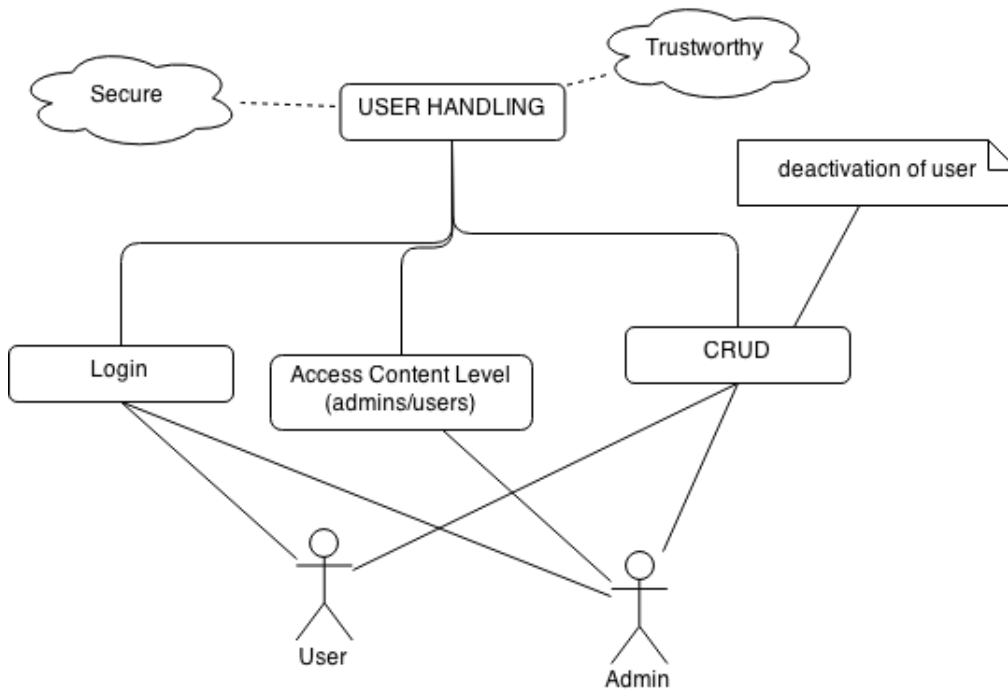
D.7 Goal model for venue management



D.8 Goal model for revenue collection



D.9 Goal model for user handling



References

- [1] 8th annual state of agile survey. <http://www.versionone.com/pdf/2013-state-of-agile-survey.pdf>, 2014.
- [2] Scott Ambler. *Agile modeling: effective practices for extreme programming and the unified process*. John Wiley & Sons, 2002.
- [3] Kent Beck. *Extreme programming explained: embrace change*. Addison-Wesley Professional, 2000.
- [4] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. *The agile manifesto*, 2001.
- [5] Lan Cao and Balasubramaniam Ramesh. Agile requirements engineering practices: An empirical study. *Software, IEEE*, 2008.
- [6] Dick Carlson and Philip Matuzic. *Practical Agile Requirements Engineering*. Technical report, 2010.
- [7] Alistair Cockburn. *Agile software development: the cooperative game*. Pearson Education, 2006.
- [8] Alistair Cockburn and Jim Highsmith. Agile software development: The people factor. *Computer*, 34(11):131–133, 2001.
- [9] M. Cohn. *User Stories Applied: For Agile Software Development*. The Addison-Wesley Signature Series. Addison-Wesley, 2004.
- [10] Anne Dardenne, Axel Van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of computer programming*, 20(1):3–50, 1993.
- [11] Martin Fowler. The new methodology. *Wuhan University Journal of Natural Sciences*, 6(1-2):12–24, 2001.
- [12] Graham R Gibbs and Celia Taylor. How and what to code. *Online QDA*, 2005.
- [13] Jennifer Horkoff and Eric Yu. Analyzing goal models: different approaches and how to choose among them. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 675–682. ACM, 2011.
- [14] E. Hull, K. Jackson, and J. Dick. *Requirements Engineering*. Springer, 2011.
- [15] Elizabeth Hull, Ken Jackson, and Jeremy Dick. *Requirements engineering*, volume 3. Springer, 2005.
- [16] Suzette S. Johnson. Requirements Engineering in an Agile Environment. (October), 2009.
- [17] Henrik Kniberg. Scrum and xp from the trenches. *Lulu. com*, 2007.
- [18] Gerald Kotonya and Ian Sommerville. Requirements Engineering : Processes and Techniques. *Star*, page 294, 1998.
- [19] Dean Leffingwell. *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional, 2010.

- [20] Dean Leffingwell. Scaled agile framework. *Siehe: <http://scaledagileframework.com>*, 2013.
- [21] Dean Leffingwell and J Aalto. A lean and scalable requirements information model for the agile enterprise. *Leffingwell LLC*, 2009.
- [22] Tim Miller, Sonja Pedell, Leon Sterling, and Bin Lu. Engaging stakeholders with agent-oriented requirements modelling. In *Agent-Oriented Software Engineering XI*, pages 62–78. Springer, 2011.
- [23] Sridhar Nerur, RadhaKanta Mahapatra, and George Mangalaraj. Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5):72–78, 2005.
- [24] Alex Norta, Msury Mahunnah, Tanel Tenso, Kuldar Taveter, and Nanjangud C Narendra. An agent-oriented method for designing large socio-technical service-ecosystems. In *Services (SERVICES), 2014 IEEE World Congress on*, pages 242–249. IEEE, 2014.
- [25] Frauke Paetsch, Armin Eberlein, and Frank Maurer. 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, 2003.
- [26] Balasubramaniam Ramesh, Lan Cao, and Richard Baskerville. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20(5):449–480, November 2007.
- [27] Colin Robson. *Real word research*. Oxford: Blackwell, 2002.
- [28] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131–164, 2009.
- [29] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons, 2012.
- [30] Alberto Sillitti and Giancarlo Succi. 14 Requirements Engineering for Agile Methods. *bilder.buecher.de*.
- [31] Ian Sommerville and Pete Sawyer. *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc., 1997.
- [32] Leon Sterling and Kuldar Taveter. *The art of agent-oriented modeling*. MIT Press, 2009.
- [33] Tanel Tenso and Kuldar Taveter. Requirements engineering with agent-oriented models. 2013.
- [34] Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249–262. IEEE, 2001.
- [35] Axel Van Lamsweerde et al. Requirements engineering: from system goals to uml models to software specifications. 2009.