CAGED 2.0: Know your enemy

*Improving the usability of an Android app*

Ida Marie Frøseth

Thesis submitted for the degree of
Master in Informatics: Programming and Networks
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2017
CAGED 2.0: Know your enemy

Improving the usability of an Android app

Ida Marie Frøseth
If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat.

–The Art of War by Sun Tzu
Abstract

This thesis improves the usability of the Communication Application with Geographical Element Data (CAGED) application, an Android application for improved Situational Awareness (SA) at the individual soldier level. CAGED was developed in 2016 as part of a demonstrator in the SMART experiment at the Norwegian Defence Research Establishment (FFI). The experiment investigated how smart technology could be used as a cheap and low-complexity platform for units with limited resources and time for training.

The results from the SMART experiment were promising, however they also uncovered several limitations with the demonstrator. This thesis continues the SMART experiment by improving the usability of CAGED. It includes an analysis of the usability requirements for the application and a study of state of the art and related work. The highest prioritized requirements have been implemented, tested and evaluated using both qualitative and quantitative methods through four user tests and one technical test.

The new design (called CAGED_2.0) enhances the data visualization, the information architecture, the user interactions and the performance. The results from the user tests indicates that the users are more satisfied with the application after the changes and they seems to agree that the usability has improved.

A new geolocation mechanism was implemented which uses sensor fusion to save power when the user is stationary. The results from the technical test showed that the new mechanism could save as much as 50% of the power while preserving the accuracy compared to the previous solution.
Preface

This thesis represents the final product of my master degree in Informatics at the University of Oslo. The work was conducted at the Norwegian Defence Research Establishment and supervised by Dr. Frank Trethan Johnsen and Cand. Scient. Trude Hafsøe Bloebaum.

This thesis has been a long journey and I would not have been able to complete it without the invaluable help and support given by various people. First and foremost I would like to express my gratitude to my supervisors Frank and Trude for introducing me to this very interesting topic and for their support, dedication and proofreading. They have allowed this work to be my own, but steered me in the right direction whenever I’ve needed it. I would also like to thank their colleagues, Marianne Rustad Brannsten and Ketil Lund for spending time and effort to ensure that my tests were successful.

Thanks to the Norwegian Armed Forces Cyber Defence (CYFOR) and the Norwegian Home Guard for allocating resources for my tests. I would also like to thank the CYFOR for giving me the opportunity to fulfill my degree and to my colleagues and friends at Jørstadmoen for being there when I have needed some social refill.

Thanks to Elisabeth Desiree Eriksen, Ola Bjørn Nordraak and Tamara Ward for helping me with the proofreading of this thesis.

Finally, I would like to express my love to my family for their help and support. A special thanks to my dear brother Gunnstein Thomas Frøseth and my mother Ingrid Hoel for motivating me and helping me in proofreading this thesis. Last but not least, thanks to my beloved Lasse Sæther and my son Iver Sæther for helping me keep a nice balance between the reality and the intense work with this thesis. I am looking forward to the coming years with them where I will help them reach their goals.

Ida Marie Frøseth

Kjeller, May 2017
# Contents

1 Introduction ........................................ 1
   1.1 SMART – pervasive situational awareness at the individual soldier level .............................. 2
       1.1.1 The Norwegian Home Guard .................. 2
       1.1.2 Titans – the project demonstrator .......... 3
       1.1.3 The SMART project main findings ............ 4
   1.2 Motivation ........................................ 5
   1.3 Problem description .............................. 5
       1.3.1 Problem statement ............................ 6
   1.4 Scope and Limitations ......................... 6
       1.4.1 Google play services issues .................. 7
   1.5 Research Methodology .......................... 8
       1.5.1 Step 1: State requirements .................. 8
       1.5.2 Step 2: State specification .................. 8
       1.5.3 Step 3: Design and implementation ........... 8
       1.5.4 Step 4: Test the system ..................... 9
   1.6 Contribution ..................................... 9
   1.7 Thesis outline ................................... 9

2 Background and Requirements analysis ............. 11
   2.1 Usability ........................................ 11
   2.2 Situational Awareness ........................... 12
       2.2.1 Specialized military SA systems ............ 12
   2.3 Mobile smart technology opportunities and limitations ........................................ 14
       2.3.1 Input ........................................ 14
       2.3.2 Output ....................................... 14
       2.3.3 Mobility ..................................... 15
   2.4 CAGED_1.0 ....................................... 15
       2.4.1 Basic functionality ........................... 15
       2.4.2 Blue force tracking in CAGED_1.0 .......... 16
2.4.3 Observations ........................................... 16
2.4.4 Technical solution .................................... 17
2.4.5 The CAGED_1.0 user test .............................. 18
2.5 CAGED_1.0 Analysis ...................................... 19
   2.5.1 Visual representation ................................. 19
   2.5.2 Information architecture .............................. 22
   2.5.3 Interactions ........................................... 25
   2.5.4 Performance .......................................... 28
   2.5.5 Summary of requirements ............................. 29

3 State of the art and related work ......................... 31
   3.1 Other military smart experiments ...................... 31
      3.1.1 Nett Warrior ......................................... 31
      3.1.2 TIGR .................................................. 31
      3.1.3 PROMISE ............................................. 32
      3.1.4 CEI .................................................. 32
   3.2 Map ...................................................... 32
      3.2.1 Map theory ........................................... 32
      3.2.2 Map data types ...................................... 34
      3.2.3 Online map services ................................. 35
      3.2.4 Map libraries for Android ......................... 38
   3.3 Data visualization ...................................... 39
      3.3.1 Material Design guidelines ......................... 39
      3.3.2 Joint Military Symbology ......................... 40
      3.3.3 Visualization techniques for geotagged data .... 42
      3.3.4 Related work ........................................ 43
   3.4 Information Architecture ............................... 44
      3.4.1 Military standards .................................. 44
   3.5 Location tracking ...................................... 44
      3.5.1 Location strategy and related terminology ...... 45
      3.5.2 Related work ........................................ 46
      3.5.3 Location tracking with Android .................. 48

4 CAGED_2.0 Design .......................................... 51
   4.1 Visualization representation ......................... 51
      4.1.1 Configurable background map from Geodata ..... 52
      4.1.2 APP-6 inspired shapes ............................. 55
      4.1.3 Clustering .......................................... 57
   4.2 Information architecture ............................... 57
      4.2.1 Menu ................................................ 58
Chapter 1

Introduction

Situational awareness (SA) is a key enabler for military sovereignty [1] and has throughout history played an important role in military operations [2]. SA is how the situations is understood by an individual and by having good SA better and faster decisions can be made.

With the ever evolving technology, new and faster ways for information sharing are made possible. How to best leverage these opportunities in military operations have been thoroughly studied. The concept of Network-centric warfare, introduced by the United States Department of Defense (USDoD) in 1999, is considered as one of the originators for this work within western military organizations. The concept was then adopted by the North Atlantic Treaty Organization (NATO) early 2005 under the name NATO Network Enabled Capability (NNEC). In 2007 it was officially adopted by the Norwegian Armed Forces when it was included in the Norwegian Armed Forces Joint Operational Doctrine under the name Network Based Defense [3].

In Network-centric warfare the organizational boundaries are removed, and sensors, decision makers and effectors from different organizational units collaborate to best solve the operation [1, 3, 4]. Communication and interoperability between units and even nations is important in this concept and imply a need for sensor and communication equipment for every single soldier. This is not achievable with traditional military technology due to cost and lack of interoperability, and has led to an increased interest for Commercial off-the-shelf (COTS) products and open standards.

Military technology is expensive due to the limited market and various special requirements like security, connectivity, robustness and usability. Since the military domain historically has been one of the main drivers for technological innovation, the tradition of using COTS and open standards within the community is limited. With a fast growing market, the
commercial domain has now taken over in many areas. One technology that is interesting within the concept of NNEC is smart technology. Smart technology offers lightweight computational devices at low cost. They typically have multiple integrated sensor and communication capabilities and this makes them strong candidates for improving SA in military operations [5–7]. However, the opportunities of smart technology often come at the expense of battery lifetime, sensor accuracy and security.

1.1 SMART – pervasive situational awareness at the individual soldier level

Several initiatives have examined the use of smart technology in military operations, and the SMART project at the Norwegian Defence Research Establishment (FFI) is one of them. The SMART project was a Concept Development and Experimentation (CD&E) activity in 2016 [6]. It differs from other military smart experiments by focusing on users with limited resources and time for training, primarily the Norwegian Home Guard (HV). The objective of the SMART project was to evaluate the operational value of using smart technology for improved SA.

1.1.1 The Norwegian Home Guard

The HV is one of five branches of the Norwegian Armed Forces and serves as a quick reaction force. Their main responsibility is territorial protection and consist of more than 45,000 soldiers distributed in 11 districts [8]. The HV is a mobilization force with 15 Rapid-reaction Intervention Forces (ISTY) and 241 Area forces. While the ISTYs are capable of rapid response and consist of highly trained and equipped personnel, the Area forces have longer reaction time, are less equipped and the personnel have less training. The SMART project aimed at providing a communication system for the Area forces, hence the concept had to be cheap and require as little training and management as possible [6].

Command structure

As for any military unit, the command structure within an Area force is hierarchical, see Figure 1.1. The smallest unit is a squad. Squads usually consist of about seven people who are the soldiers on the ground. Each

---

1Mobilization force means that the soldiers are not employed by the military on a daily basis, but train on a regular basis.
squad member has a designated task and is led by a squad leader. Three to five squads are organized in one platoon, and three two five platoons are organized in one Area. Each organizational level has one commander and one second in command. Additional roles like intelligence, operations, logistics, plans etc. are found at the Area level and up.

The HV soldier is a person who is 18-55 years old, and has at a minimum fulfilled one year of mandatory conscription [9]. This applies for both soldiers on the squad and platoon level. The commanders at each level usually have more training than the average soldier.

1.1.2 Titans – the project demonstrator

Through the SMART CD&E a demonstrator called Titans was developed, see Figure 1.2. Titans included a Representational State Transfer (REST) back-end called Athena, a web-based management interface called Metis and an Android client application called Communication Application with Geographical Element Data (CAGED). Third-party software was
also used to enable text based instant messaging and to add transmission confidentiality with Virtual Private Network (VPN).

The CAGED application was the main user interface for the squad and platoon members and enabled them to build a common understanding of the situation. The application consists of a map where all the other users’ locations are displayed. Various observations added by the users are also shown, see Figure 1.3. The users could communicate through observations by writing comments or adding pictures. The location of a user was logged with the smart device Global Positioning System (GPS) sensor, and the data was shared among the user through the REST server.

Figure 1.3: CAGED main user interface

1.1.3 The SMART project main findings

To evaluate the operational value of using smart technology, Titans was tested by one HV Area force during a regular military exercise. The commanders and the second in command at squad and platoon level were deployed with the CAGED application. The Area command had the management interface Metis. The users’ satisfaction and attitude were measured through questionnaires. The results were promising and show that the use of smart technology can provide a cheap and easy-to-use communication alternative for units similar to the HV [6], see Figure 1.4.

During the SMART experiment, it was discovered that certain areas of the Titans demonstrator were in need of improvement. Limitations with the security, usability and robustness were found [6].
Figure 1.4: The users’ attitude towards and satisfaction with CAGED, results from the user test questionnaire.

1.2 Motivation

The objective of this thesis is to improve the usability of the CAGED application. Improving usability of the application is essential to further demonstrate that COTS is a viable option to the alternative. Improved usability also encourages other branches of the military to adopt the technology, which in turn generates more resources to address other limitations of COTS, e.g. security, connectivity and robustness.

This thesis uses the following identifiers when later referring to the demonstrator used during the last user test conducted by the SMART experiment:

- Titans_1.0 - refers to the entire system.
- Metis_1.0 - refers to the management interface.
- CAGED_1.0 - refers to the Android SA application.
- Athena_1.0 - refers to the REST back-end.

1.3 Problem description

The SMART project did not do any analysis of the issues other than identifying the following as the most critical limitations of the CAGED_1.0 application:

<table>
<thead>
<tr>
<th>Limitation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very dissatisfied vs very satisfied</td>
<td>5.38</td>
</tr>
<tr>
<td>Very displeased vs very pleased</td>
<td>5.24</td>
</tr>
<tr>
<td>Very frustrated vs very contented</td>
<td>5.34</td>
</tr>
<tr>
<td>Very terrible vs very useful</td>
<td>5.57</td>
</tr>
<tr>
<td>Very impractical vs very good</td>
<td>6.00</td>
</tr>
<tr>
<td>Very bad idea vs very good idea</td>
<td>5.47</td>
</tr>
<tr>
<td>Very difficult vs very easy</td>
<td>5.72</td>
</tr>
<tr>
<td>Very problematic vs very intuitive</td>
<td>5.79</td>
</tr>
</tbody>
</table>
- The level of detail in the map
- Imprecise GPS positioning
- Severe battery drain

An analysis of CAGED_1.0 was first conducted in this work to get a better grasp of the already identified issues and to potentially uncover additional problems. The analysis ended in a prioritized list of requirements. The highest prioritized requirements were investigated, implemented and tested and compared with the results form the SMART project user test.

1.3.1 Problem statement

The derived problem statement was as follows:

1. What are the main usability requirements of the CAGED application?
2. How to best solve these challenges with the known limitations of smart technology using state of the art and applying them to Android?

1.4 Scope and Limitations

The usability issues of the SMART demonstrator are both related to Athena_1.0 and CAGED_1.0. Due to the limited time, this thesis focuses on the usability issues of the CAGED_1.0 application. It does not consider the communication with the server or the choice of back-end system (including the choice of using Java, REST or a Postgres database). It will neither go into details about the choice of using native Android versus a cross-platform framework or any other platform. For readers not familiar with native Android development, an overview is given in Appendix A.

In the remainder of this thesis, any issues discussed are to be considered issues with the CAGED application and not the other demonstrator components, unless otherwise stated.

Security can have a large impact on the usability of a system, for example by having security mechanisms that make the system difficult to use. However due to the limited time available, security has been defined as out of scope. Another reason for leaving out security is that security of smart technology for use in military operations is well covered by Mancini in [10].
With smart technology there are a lot of opportunities and challenges related to a military SA application and all of them cannot be covered within a single thesis. Several issues are addressed but only a subset of them has been further investigated and solved as part of this work. What have been done is a new location strategy, a better map, better data visualization and changes to the information architecture.

The minimum Android version which will be supported by the application is Android 5.0. This release represents major updates to the Android system and especially the user interface which was rebuilt based on the material design language (for more details about material design see Chapter 3.3.1) [11]. The release also improves the power management and power consumption compared to previous releases [11].

A bachelor group from the Norwegian University of Science and Technology (NTNU) was working in parallel with this thesis to improve Metis_1.0. Their work resulted in a completely new implementation of Metis: Metis 2.0 [12]. Several severe limitations with the back-end, Athena, had been discovered. Hence, a new design and implementation was needed for the back-end. The work with Athena_2.0 was scheduled as a collaborative effort, since it was expected to impact both CAGED_2.0 and Metis_2.0 design and development. An overview of the most severe limitations of Athena_1.0 and the design choices done for Athena_2.0 is included in Appendix B.

1.4.1 Google play services issues

Google play services offers Application Programming Interface (API)s and services on top of the Android framework. They reduce the complexity for the developer and offers access to data gathered and owned by Google. The APIs are known to be good and easy to use, and in some cases they are the recommended approach in the Android documentation [13]. Even though several of the APIs are strong candidates to be used in CAGED they cannot be used due to the license.

Google play services offers a free license in most cases, however through the license the developer and user have to agree with Google’s terms of Service, privacy policy and their collection of user data. This includes collecting the user location, device type and local storage information [14]. In an application like CAGED this is sensitive data which should not be shared with any third party.
1.5 Research Methodology

This work falls within Denning’s Design paradigm [15].

Denning describes computer science as an interdisciplinary field with three different paradigms [15, 16]: Theory, Experimentation and Design. While the first two paradigms are concerned with building broad conceptual frameworks and models within the entire field or a domain [15], the third paradigm is about Constructing computer systems that support work in given organizations or application domains [15].

Design is usually performed as an iterative process with the following four steps [16]:

1. State requirements.
2. State specification.
3. Design and implement.
4. Test the system.

Four iterations have been performed as part of this thesis. For each iteration adjustments were made both to the requirements and the design, and this has led to a final design – referred to as CAGED_2.0.

1.5.1 Step 1: State requirements

During the first phase, CAGED_1.0 was analyzed through an Heuristic Evaluation (HE) and by analyzing the data from the questionnaires collected by the SMART project during the user test. By combining the HE and the user data, it was possible to uncover additional issues related to the information architecture. The analysis ended in a prioritized list of requirements, and is covered in Chapter 2.

1.5.2 Step 2: State specification

Based on the priority of the requirements found in step 1, a subset was chosen and a study of related work and state-of-the-art was done. This study formed the foundation for the specification of a new design, and is covered in Chapter 3.

1.5.3 Step 3: Design and implementation

Based on the findings from step 2 a new design was proposed and implemented, see Chapter 4. The changes include changing the map,
redesigning the menu and implementing a power balanced location tracking strategy (see Chapter 3.5.2 for a definition of a location strategy).

1.5.4 Step 4: Test the system

One technical and four user tests have been performed to evaluate the developed application. The technical test was a quantitative test to evaluate the new location strategy on accuracy, speed and power consumption. While the user tests evaluated the usability through questionnaires, interviews and observations. Chapter 5 covers the findings of the technical test and the last user test. The first three tests have been used to improve the design to a final product, and a summary of the findings from these test are included in Appendix C, D and E.

1.6 Contribution

This thesis contributes with an analysis of the usability of the CAGED_1.0 application and a solution to the most severe limitations. An improved version of the application is developed which can be used for further experiments with smart technology in military operations. The improvements include a better map, reducing the power consumption, making the localization more accurate, better data visualization and improving the information architecture.

The work also contributed to a new and improved version of the backend. The work with Athena_2.0 was done in collaboration with a bachelor group from NTNU.

1.7 Thesis outline

The rest of this thesis is organized as follows:

Chapter 2 provides the background for the work by first defining some basic terminology. Thereafter, details about issues related to mobile smart technology is covered, before an analysis of CAGED_1.0 is included. The chapter is summarized with an overview of the uncovered requirements and a statement of which are solved within this thesis.

Chapter 3 covers state-of-the-art and related work to the chosen requirements from the previous chapter.
Chapter 4 describes the design of the enhanced solution.

Chapter 5 evaluates the enhanced solution and includes the result from the technical test and the last user test.

Chapter 6 concludes the work.

Chapter 7 proposes future work.
Chapter 2

Background and Requirements analysis

The main goal with this thesis is to improve the usability of the CAGED_1.0 application. In order to achieve this, an understanding of what is meant by usability is needed. Hence, this chapter start by defining usability. A definition of SA follows, since CAGED is an application for improved SA. To give the reader an understanding of what is expected by a military SA application, an overview of a specialized military SA system and its characteristics is given. Then, well-known strengths and limitations of smart technology is covered, before looking at CAGED_1.0 in more detail. The chapter is concluded with an analysis of the CAGED_1.0 application and summarized with a prioritized list of requirements for a new design.

2.1 Usability

This thesis uses the definition for usability found in the International Organization for Standardization (ISO) 9241-11 Guidance on usability [17]: Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. Or said with simpler words by Rubin et al. [18]: the user can do what he or she wants to do the way he or she expects to be able to do it, without hindrance, hesitation, or questions.

Usability is more than ease-of-use [19]. It is about how effective the user achieve his/her goal with correctness and ease of use. How efficient or fast the goal is achieved and how much the user enjoy using the system (satisfaction) [18, 19].
2.2 Situational Awareness

Situational Awareness (SA) is how an individual understands the current situation, and involves collecting, combining and interpreting data [20]. By having good SA, the individual can make better decisions faster. In military operations, multiple individuals collaborate to solve a mission. By having a common understanding of the situation, synergies can be achieved. Furthermore, if multiple people collaborate to build a common SA, more insight can be gained since more information is available.

Endsley [21] describes three different levels of SA namely perception, comprehension and projection. The first level is the fundamental level, which encompass the perception of sensor cues. The second level is concerned with a higher level of understanding where multiple perceived cues are combined and interpreted and their relevance to the person’s goal is determined. At the highest level of SA, the person is able to predict the future, and this level is strongly connected to a person’s experience [21].

It is important to notice that situational awareness is individual and depends on how information is perceived and interpreted by the user [20]. Too much information can lead to confusion if the person is not able to consume it. Whether information is relevant depends on multiple factors including the situation, the role of the one using the information and his/her level of experience. For soldiers on the ground, relevant information usually includes the different actors in the area, the terrain, the weather, the available infrastructure and different incidents.

Technology to support SA in military operations has existed for a long time and often includes a map due to the importance of the physical environment.

2.2.1 Specialized military SA systems

Military SA applications are typically part of a larger Command and Control Information System (C2IS) and usually includes Blue Force Tracking (BFT). A blue force in a military context is a friendly / allied force. The name BFT underlines the importance of avoiding blue on blue / friendly fire by tracking the location of friendly forces and displaying their location on a map. The tracking is often done with GPS and is shared through various data connections. The type of connection vary a lot depending on the location and the mobility of the unit, but it is often a Mobile ad hoc network (MANET) using military radios with encryption.

In addition to tracking friendly forces, SA applications often serve a
role in tracking hostile and neutral forces. This is usually done by units reporting observations and manually plotting their location. An example of an operational SA application is the Norwegian Battlefield Management System (NorBMS) by Teleplan used by the Norwegian Army.

Typical characteristics of a specialized military SA system follows.

**Highly secure**

Specialized military SA systems have typically highly secure mechanism so they can carry classified data.

**Robust**

The devices used in a military SA system are usually rugged to survive the harsh physical environment in military operations. The communication is also more robust than commercial services. The radios often have mechanism, like frequency hopping and spread spectrum, to mitigate electronic warfare attacks. An advantage with building a MANET with standalone radios, is that it is not dependent upon civilian infrastructure.

**Low data capacity and high delay**

Radio communication is limited by physical laws, and the Shannon-Hartley theorem says that the data capacity is relative to the frequency bandwidth of a signal – higher bandwidth results in higher data capacity. Furthermore, the range of the radio signal depends on the frequency and power – lower frequencies have typically longer range, and higher frequencies are more dependent upon line-of-sight. In other words, military radios often use lower frequencies to be able to reach longer and to be more robust, but this reduces the data capacity and increases the delay.

**Integration**

The integration between components in a specialized military SA systems is usually not as streamlined as in smart technology. The soldier typically has to carry one physical GPS, one radio and one computer/laptop [5].

**Requires training**

Specialized military equipment often requires configuration and set-up to enable the different components to communicate with each other, which is often a source of error. Furthermore, specialized SA applications are
often made for the trained soldier and military conventions and standards are used. These standards are effective in their purpose, but often requires training to fully understand and utilize. An example is the **Allied Procedural Publication 6 (APP-6)** standard which is covered in Chapter 3.3.2.

**Expensive**

Due to all the previous mentioned characteristics, specialized military SA systems are custom built. Since the market is relatively small compared to the civilian market, the equipment is expensive. For example can a rugged laptop be around ten times the price of non-rugged computer with the same computing capacity.

### 2.3 Mobile smart technology opportunities and limitations

Mobile smart technology represents both opportunities and limitations from a usability perspective. This section gives an overview of the most central of them. They are grouped into input, output and mobility and the information is mostly extracted from the work of Huang [22] and Veldhuis [23].

#### 2.3.1 Input

Smart devices are usually small sized and touchscreen based, which can lead to problems like fat-fingers (the fat-finger problem refers to the problem with precise input on touchscreen based devices). The touchscreen is sensitive to environmental impact like cold and rainy weather, but it also enables new ways for interaction like multi-finger gestures. Smart devices usually come with additional sensors and input facilities and can support multimodal interactions (input from multiple sources like shake, voice and touch). Examples are the headset remote control, voice control using the audio input and an accelerometer, magnetometer and gyroscope to detect activity or gestures.

#### 2.3.2 Output

Smart devices can come in different forms and sizes. To support this variation of size within a single application is in itself a challenge [24]. Especially small screens are troublesome and problems related to hierarchical
and hidden navigation get more prominent. With limited screen size, there will always be a trade-off between the amount of information and the usability. On the other hand, since smart technology comes with a range of sensors, it is possible to use techniques like Augmented Reality (AR) and Virtual Reality (VR) for better data visualization. In addition to a screen does smart devices usually come with other output facilities like an audio output and a vibrator. These interfaces are often used to notify the user or enhance the user experience with feedback, known as tactile queues. An example of a tactile queue is by using the vibrator to indicate that a button is pressed.

2.3.3 Mobility

To support mobility, smart devices are small and light weight and this may come at the expense of battery, computational resources and sensor accuracy. Furthermore, smart phone applications have to cope with the user being on the move. This includes handling disrupted network connections as well as facilitate interaction during movement by reducing the cognitive attention required by the user [23, 24]. On the other hand, smart devices usually support multiple types of network connections which can be used in different types of situations.

Due to the nature of military operations, several of the listed limitations are magnified when developing military applications. Examples are extreme weather conditions, the limited time to interact with the system due to dangerous situations and unstable and unreliable infrastructure.

2.4 CAGED_1.0

This chapter gives an overview of the CAGED_1.0 application. It starts with the basic functionality of the application before looking at the technical solution. A short section about the SMART project user test is also included since the analysis in the next chapter uses the data from this test.

2.4.1 Basic functionality

The CAGED_1.0 application is, as already mentioned, an application for improved SA at the individual soldier level. It has the basic functionality as found in most specialized military SA applications like BFT and the
possibility to report hostile, friendly, unknown and neutral assets. These assets are hereafter called observations.

2.4.2 Blue force tracking in CAGED_1.0

A user’s location is contentiously logged by the application with the device built-in GPS and sent to the server over the Internet. A timestamp is added to the location so it is possible to decide the age of the location. Both the user’s own location and other users’ locations are displayed on the map. While the user’s own location is displayed using a blue circle, the other users’ locations are displayed with a drop shaped icon, see Figure 2.1a. If a user location is older than two minutes, he/she is considered inactive. This is visualized by changing the the color of the icon to gray, see Figure 2.1b. If multiple markers are close together, they are clustered to avoid information overflow. A blue circle with a number is used as the icon for a user cluster. The number indicates the number of users inside the cluster. If a user cluster is clicked, a list of all the users within the cluster is revealed.

(a) New user location.  
(b) Old user location.

Figure 2.1: Other users’ location is displayed using a drop.

2.4.3 Observations

An observation in CAGED_1.0 is visualized using a circular marker with an icon inside, see Figure 2.2. The icon indicates what the observed assets is, e.g. car, soldier or road-block, while the background color of the marker indicates the status. Red is used for hostile observations, blue for
friendly, green for neutral and orange for unknown. It is also possible to reveal more information or add comments and files to an observation by clicking on it. Observations are clustered when multiple observations are located close together. Clustered observations are displayed with a black circle with a counter, see Figure 2.2.

To register an observation the user has to click and hold the map at the location where he/she wants to put the observation. This will reveal a dialog window with a scheme as shown in Figure 2.3. Besides giving the observation a title (Tittel), set the status and the icon (Ikon), the user can add a description (Beskrivelse) and one or more files (LEGG TIL FIL). When the user saves the observation (Lagre), it is sent over the Internet to the server. If there is no network connection, the observation is stored locally until the network is available.

![Figure 2.3: CAGED_1.0 add observation dialog](image)

### 2.4.4 Technical solution

The CAGED_1.0 is a native Android application which is supported by devices with Android 4.4 and onwards. The application is built on top of the open source OpenStreetMap (OSM) Droid library, a library to simplify the development of map based applications. OSM Droid supports multiple types of maps. The CAGED_1.0 uses the default map, which is the free OSM. See Chapter 3.2 for map theory and map based application development in general and the OSM Droid library specifically.
Synchronizing the situation

Military operations are dynamic, and it is important that the information in an application like CAGED is up to date. Since the system is realized using client-server model, the user has to poll the server regularly for new information. To limit the amount of data transferred, only new or updated information is sent from the server to the client.

The poll based synchronization of the situation, requires that the device has a connection to the Internet. For units located outside of reach of a network connection, data can be shared between two devices directly using Android Beam. Android Beam is a Near Field Communication (NFC) technology with a range of a couple of centimeters. Hence, the device to device communication were used in a store-carry-forward manner.

Security

Security in CAGED_1.0 is both added on the application and the network layer. Authentication is used on the application layer to restrict access to information. The user can either be authenticated using username / password or a NFC card and PIN code. Confidentiality is added on the network layer by using OpenVPN with pre-shared certificates. OpenVPN supports multiple security modes, but for the Titans_1.0 it was used with AES-128 encryption for confidentiality and SHA-256 for integrity. The advantage of adding the security on the network layer is that all the data is encrypted between the client and the server, hence all third party applications are also secured within the same tunnel.

2.4.5 The CAGED_1.0 user test

The SMART project user test was conducted by having 33 HV soldiers from the Area forces using the CAGED_1.0 application as a tool during a regular exercise. The test was executed as follows:

1. A 20 minute brief was given as an introduction to the project and the application before the users started using it. During this presentation, no restrictions on how to use the system were given.

2. The users answered the questionnaire part I.

3. The devices were handed out.

4. The soldiers used the application during a five days long exercise

5. The users answered the questionnaire part II.
Before and after the test, the users were asked to answer a questionnaire, referred to as questionnaire part I and part II. In the questionnaire, the users were asked both closed-ended matrix questions and open-ended questions. The intent of the part I questionnaire was to uncover the users experience, habit and attitude towards using smart technology, as well as their expectation to the developed system. The part II questionnaire gathered data about the experience from the actual usage and what potential the users saw in smart technology [6].

2.5 CAGED_1.0 Analysis

The analysis in this chapter is a qualitative study based on two sources, an analysis of the data gathered by the SMART project and a HE. By using two methods the validity of the findings are increased [25]. It can also help to compensate for only having one evaluator during the HE. HE should be performed independently by three to five persons as pointed out by Nielsen and Molich in [26].

A HE is an informal method to evaluate the usability of a user interface design. It is conducted by simply looking at the interface and evaluate it based on some chosen design principles (heuristics) [26]. The advantage of the technique is that it is effective, can be used early in the development phase and does not require much resources [27, 28]. The method also has some shortcomings as it could be biased by the current mindset of the evaluators [26]. Nielsen’s 10 usability heuristics [29] are used in the HE of CAGED_1.0. The 10 heuristics are included in appendix F for readers who are not familiar with them. Only the main findings from the HE is included in this chapter, see Appendix G for the complete evaluation.

The analysis that follows is organized in the following four categories: visual representation, information architecture, interactions and performance. Each category states several issues and derived requirements with priority. At the end of this chapter a summary of the findings is included to give the reader an overview of the identified open issues.

2.5.1 Visual representation

Map fidelity

The map used in CAGED_1.0 is OSM [30]. OSM lacks information in some areas, and the users mentioned contours, roads and building as examples of information which were not present. Multiple users mentioned the
Norgeskart app [31], which uses Kartverkets open map database [32], as an example of a good map. The derived requirement is as follows:

**Requirement 2.1.** The map in CAGED needs fidelity similar to Norgeskart 1:50000 map or better.

The consequences of a map with low fidelity might be navigation errors, lack of trust in the system and that the user rather uses other mapping applications for navigation. The consequences are crucial for the application to be used therefore this requirement is assigned a **high** priority.

**Map provider**

CAGED_1.0 serves the map from an offline directory. However due to limited storage and the size of map data (more on map size and storage in Chapter 3.2), it is only possible to store a limited area. In CAGED_1.0 this is solved by pre-downloading the defined area of operation. However, if the user needs to operate in unforeseen areas or the area is too large to be contained within a file, the user will not be able to get a map. OSM has also changed their policy so it is no longer allowed to pre-download large map areas [6]. Without a map the application is almost useless, hence the following requirement should be given a **high** priority:

**Requirement 2.2.** The user should be able to get a map outside its area of operation on-demand.

**Observations — color and shape**

Observations of different type of actors (friendly, hostile, neutral and unknown) are displayed using different colors but with the same shape (circle). This would be a problem for users with color deficiency, and in some cases they would not be able to distinguish a hostile from a friendly observation, see Figure 2.4. This was also notified by a user in the open-ended question who said it was difficult to differentiate between users and observations in the map view (clustered users are also displayed with a circle).

Since this issue can make the system useless for special groups of users the following requirement should be given a **high** priority:

**Requirement 2.3.** The application should use a secondary attribute (beside color) to distinguish different types of actors.

---

1 The online tool [http://www.color-blindness.com/coblis-color-blindness-simulator/](http://www.color-blindness.com/coblis-color-blindness-simulator/) was used to create the colorblind image.
Figure 2.4: The colorblindness issue: Figure 2.4a and 2.4b show two hostile, one friendly, one unknown and two neutral observations with and without colors.

Clustering leads to SA issues

In CAGED 1.0 clustering is used to avoid information overload. This makes the view less messy, but since different types of observations are clustered together and with the same type of marker (circle) and color (black), it is impossible to distinguish between dangerous and safe areas without having to interact with the application, see Figure 2.5.

Figure 2.5: Clustering problem - the left cluster consist of only friendly, the right only hostile and the bottom a mix of hostile and neutral, but it is impossible to tell without interacting with the application.

The clustering is also too aggressive since it clusters down to the largest zoom level. If two observations are very close it is not possible to see what type of observation is inside the cluster without clicking on the cluster and reveal a list of observations. One user even asked for a feature to
disable/enable the clustering functionality in the open-ended questions. This leaves the following requirement with medium priority:

**Requirement 2.4.** It should be visually easy to spot what types of actors (hostile, friendly, unknown and neutral) are observed in an area without having to interact with the system.

### 2.5.2 Information architecture

Information architecture can be defined as *The art and science of shaping information products and experiences to support usability, findability and understanding* [33].

**Menu**

The menu in CAGED 1.0 is exposed through a Floating Action Button (FAB) located in the lower right corner of the map view, above the "zoom to my location"-button. The strength of the menu implementation is it requires little space, however the choice of icon and the three level hierarchy is not good.

The platform conventions for navigations says: *Focus attention on important destinations by displaying them in tabs or in the side navigation, and de-emphasize inessential content by displaying it in less prominent locations* [34]. CAGED 1.0 follow this design pattern by offering the zoom to my location button and hiding other actions in the menu.

The main menu icon is a cog icon, which is often used as a settings menu icon. This can prevent the user from finding relevant actions since it is not perceived as a menu button, see Figure 2.6. The material design guidelines recommend the use of three vertical dots to indicate a menu button [35] (see Chapter 3.3.1 for more on the material design guidelines).

![Figure 2.6: Main menu button](image)

By not following platform conventions the CAGED application can be harder to learn for new users. Furthermore by hiding functionality in a three-level menu hierarchy the user interface is not as effective as it could be. This is not a critical part of the application since the main functionality
is exposed through the map view. The following requirement is given a **medium** priority.

**Requirement 2.5.** The menu in CAGED should follow platform conventions and limit the number of interactions by removing one level of the hierarchy.

### Ordering of the observation list

![Old observation list](image)

Figure 2.7: Old observation list

The list over all the observations is not ordered chronologically and the items does not contain any timestamp of when they were last updated, see Figure 2.7. This makes updated information hard to find, as indicated by some of the user’s feedback in the open ended questions (the feedback is freely translated from Norwegian):

- **Problems with CAGED:** *It was difficult to find updated information.*

- **Ideas:** A log of the last updated information in a chronological order, and preferable the possibility to filter this list by a geographical area.

- **Ideas:** A news-feed with the last events.

- **Ideas:** A feed with a timestamp to see when an observation is added.

- **Ideas:** An easier way to find what observation I added and what information has been updated

By having a timestamp on the items in the all observation list and ordering the list by when they were updated, the user can easier find the last updated information. This will contribute to increase the user’s situational awareness, which is a core goal of the application, hence the following requirement should be given a **medium** priority:
Requirement 2.6. The observations in the list over all observations should be ordered chronologically by the time they were last updated. The items in the list should also contain a timestamp so it is easy to see how old the information is.

Military standards and conventions

There are two military standards identified as relevant for the CAGED application. The Military grid reference system (MGRS) standard and the APP-6 standard.

MGRS is an earth coordinate system used by NATO for referencing a point on earth (for a full definition of an earth coordinate system see Chapter 3.2). The CAGED_1.0 app uses latitude and longitude, but the soldiers are used to the MGRS and it is also the coordinate system displayed on their physical map. Since the squads still uses physical maps to navigate and the operation order usually refers to locations in MGRS, the efficiency and synergy by using the application is not as high as it could be. Some users even went as far as stating in the open-ended questions that the application is useless without MGRS.

By using the standard it would make the soldier more effective, hence the following two requirement should be given a medium priority:

Requirement 2.7. The location of an observation or user should be displayed using the MGRS format.

Requirement 2.8. The map should contain a MGRS grid so it is possible to find a location given over some other media.

The other standard, the APP-6, is a standard for graphically representing military units, installations, activity or other item of military interest. CAGED_1.0 does not use the APP-6 standard and this is a design choice. The argument is that the APP-6 standard requires time to learn, hence it may not be appropriate in an application aimed at the Area forces. On the other hand is the APP-6 developed through years of research and for example would the issue with color blind users, see Requirement 2.3, have been avoided if the application had used the standard. Another advantage with using the standard is that it would make the Titans easier to integrate with other military systems.

Based on the fact that the APP-6 standard require training, and only some of the soldiers in the Area forces have this training, implementing the APP-6 standard may not improve the usability for all the users. However it...
will most likely improve the usability for soldier with the training. Hence, the following requirement is assigned a low priority:

**Requirement 2.9.** It should be possible to toggle between the APP-6 standard and a more easy-to-use icon set in the CAGED application.

**Filtering**

Multiple users mentioned that they had problems filtering out relevant information, and this is despite the short time frame the app was used (with longer exercises more and older information would be present). In CAGED 1.0 the idea is that an administrator (typically a user of Metis) should remove information when it gets irrelevant [37]. However in conversations with different users it is clear that different users with different roles have a different perception of what relevant information is and is not. For example the S-2 (Intelligence) personnel of the Area Commands would need *historic* information in order to analyze how the situation is evolving, while a soldier in a squad would only be interested in the information that can influence the current situation [6]. On the other hand, it is obvious that information left out (filtered) without the soldier being aware of it can be dangerous. Hence, a filtering mechanism has to be user initiated and not necessary remove the information completely from the map, but make it visually less important.

Whether information is relevant or not depends on multiple factors, and using only time may not be adequate; An enemy observed passing in a vehicle may only be valid for hours or minutes, while information about the location of a command post would be relevant for maybe weeks. The user should therefore be able to define liveness of an observation individually which leads to the following low priority requirement:

**Requirement 2.10.** The application should support user initiated filters which makes it visually easy to distinguish between important and less important information.

2.5.3 Interactions

**Fat finger problem**

To register an observation in CAGED 1.0 the user has to press and hold at the location where he/she wants to put the observation. The clicked location will be the location of the observation and it is not possible to edit this location later. To be accurate when registering an observation is
difficult due to the fat-finger problem, and the recorded location would actually be vendor specific – the implementation of the screen hardware and how they fire touch events are vendor specific. The accuracy of the observation position will also be relative to the zoom level and the screen density. The zoom level will change the map scale and for example are 100 meters fewer pixels at a 1:500.000 scale than at a 1:10.000.

The following requirement is given a medium priority.

**Requirement 2.11.** It should be possible to give an accurate position of an observation, independent of screen size and vendor.

**Users limited time to interact**

![Figure 2.8: Answer to the question: Why didn’t you share information through CAGED, strongly disagree (1) vs. strongly agree (7) (mean value)](image)

Soldiers operate in dangerous and dynamic environments and their time to interact with their Command and Control (C2) equipment can be rather limited. This is reflected in the user feedback where the users answered it took focus away from my main task, as one of main reason for not adding observations, see Figure 2.8. This leaves the following requirement with medium priority:

**Requirement 2.12.** The main functionalities like adding observations and getting the most important situational information should require a minimum of the user attention and interaction.
Notifications

Multiple users called for a feature to be notified when there is a change in the system. Without any notification the users have to check the map for new or updated information periodically. This brings the soldiers focus away from their main task, and is both inefficient and dangerous. If the users are notified when there are new or updated information they will get a better and faster situational awareness. At the same time, if the users are constantly notified for every small change it can draw the users attention away from the main task. In other words, the user should only be notified when the information is relevant to him/her.

Some of the users’ comments regarding notifications were as follows (freely translated from Norwegian):

• *I miss a feature to be notified when something new has happened*

• Ideas for CAGED: “Pling” / Notification when there is new information

• Ideas for CAGED: Push notification

• Ideas for CAGED: Push notification when someone in my own platoon adds an observation

• Ideas for CAGED: Push notification when new an observation is added

A notification functionality was implemented and tested earlier in the development phase of CAGED_1.0, but was removed before the user test due to a bug which caused some types of phones to crash when interacting with the notification. The derived requirement is as follows:

**Requirement 2.13.** The user should be notified whenever relevant information is added or updated. A definition of what relevant information is has to be established. It can depend on the physical distance from the user, if the user has marked the observation as a favorite and if the user is the creator or editor of an observation with updated information.

The above requirement will give better situational awareness to the user faster, which is a core goal with the application. At the same time it is not crucial for the usability of the application so the requirement is given a medium priority.
2.5.4 Performance

Power consumption

About one out of three users commented in the open-ended questions that their device quickly ran out of power. The devices were fully charged and each squad were given a powerbank at the start of the exercise. The duration of the test were also relatively short compared to a real world scenario, about 40 hours. The soldiers can be days without getting access to a facility with power. Often squads use vehicle for transportation, but since they were not provided with a charger for vehicles, they were not able to charge the device.

The experienced high power consumption is not necessarily caused by the CAGED_1.0 app, but can be caused by one of the third party applications, an application installed by the user or a special usage pattern. Before the user test the SMART project found that the network traffic caused by the application as one of the main source for battery drainage \[37\]. This was improved by implementing version control which reduced the network traffic, thence the power consumption.

The reason for the power usage is not analyzed in detail, but network communication and actively listening for GPS updates are usually two power intensive sources.

The consequences of not having a device with power is crucial, and the following two requirements should have a high priority.

**Requirement 2.14.** The CAGED application should minimize the power consumption.

**Requirement 2.15.** The user should not be dependent on returning to a facility with corded electricity in order to recharge the device.

Unstable GPS location

Multiple users complained that their and other user’s location were jumping around even though the user was not moving. A jumpy or inaccurate GPS position can potentially lead to lost of trust to the system and navigation errors. The following requirement is given a high priority.

**Requirement 2.16.** The location must be stable and accurate enough to keep the trust to the system, and converge fast enough to be used as navigation.
2.5.5 Summary of requirements

The requirements are summarized in Table 2.1. The medium and high prioritized requirements are furthered investigated and improved in this thesis. The requirements 2.14 and 2.15 regarding power consumption will not be completely solved, but indirectly be enhanced through a power balanced location strategy. Most likely can the application be optimized on power in other ways as well, e.g. by reducing the network access, but this is not investigated further due to the limited time of this thesis.

<table>
<thead>
<tr>
<th>Category</th>
<th>Priority</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual representation</td>
<td>High</td>
<td>2.1 Map fidelity</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>2.2 Map provider</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>2.3 Observations - color and shape</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>2.4 Clustering</td>
</tr>
<tr>
<td>Information Architecture</td>
<td>Medium</td>
<td>2.5 Menu design</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>2.6 Ordering of the observation list</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>2.7 Military standards (MGRS)</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>2.9 Military standards (APP-6)</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>2.10 Filtering</td>
</tr>
<tr>
<td>Interactions</td>
<td>Medium</td>
<td>2.11 Fat finger problem</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>2.12 Time to interact</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>2.13 Notifications</td>
</tr>
<tr>
<td>Performance</td>
<td>High</td>
<td>2.14 and 2.15 Power consumption</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>2.16 Stable GPS routine</td>
</tr>
</tbody>
</table>

Table 2.1: Overview of the open issues in CAGED
Chapter 3

State of the art and related work

This chapter serves as the foundation for the design choices done in the new implementation. The chapter starts with an overview of other military smart experiments. Then map theory is covered, before data visualization, information architecture and location tracking.

3.1 Other military smart experiments

As already mentioned in Chapter 1, this work and the SMART project is not the only initiatives looking at how to apply smart technology for improved SA in military operations. In this section a selection of other openly published initiatives are included which have been used as an inspiration for CAGED_2.0.

3.1.1 Nett Warrior

Nett Warrior is an Android SA system designed for the United States army. It differs from this work and the SMART project by only using the device as a computation and sensor device. All the built-in wireless communication technology in the device is disabled and the device communicate through a military radio via an Universal Serial Bus (USB) connection. They also uses a customized version of the Android operation system.

3.1.2 TIGR

The Tactical Ground Reporting System (TIGR) is a cloud based system for information sharing at patrol level [38]. The system has been used in
military operations in Iraq and Afghanistan. It is a cross platform system and uses a web client as the user interface. Distributed servers were used to remove the risk of single-point of failure.

3.1.3 PROMISE

PROject Multi-touch Information System Exchange (PROMISE) is a Dutch CD&E activity from 2015 investigating the potential and maturity of smart technology in military operations. They developed a complete C2 system using commercial smart phones and available apps and found that the technology provides an added value for operational effectiveness. They expect the system to be certified at restricted level. In contradict to the SMART project, they focus on trained personnel. However, their conclusion emphasize that by using smart technology, less training is required.

3.1.4 CEI

In describe a new class of C2IS where SA is created using social networking, gamification and commercial available technology. They developed a prototype which they called Collective Environment Interpretation (CEI) and characterized as Twitter with a map. In CEI, all the users could and was encourage to contribute to improve the information.

3.2 Map

The map is a central part of the CAGED application and developing a map based application touches the field of geoinformatics. Geoinformatics can be defined as the art, science or technology dealing with the acquisition, storage, processing, production, presentation, and dissemination of geoinformation. This chapter will describe essential concepts and terminology in geoinformatics as well as the support for map data rendering on the Android platform.

3.2.1 Map theory

The earth is a three dimensional shape, and is often described as a spheroid (a three dimensional shape of an ellipse). The earth is not a perfect spheroid due to gravitation and local variations like mountains and valleys. Different spheroids can be chosen to best represent a specific geographic area, hence different models exists for different areas and purposes. To be
able to reference a point on the surface and compute distance, direction and area, an earth mathematical model is needed. An earth mathematical model includes a coordinate system, a datum and a projection.

**Coordinate system**

An important attribute of a earth mathematical model is a coordinate system with an established origin. The geographic coordinate system is an example of a world wide coordinate system, with the Greenwich meridian and equator as origin \[42\]. A location is referenced in latitude in degrees northbound from the equator and in longitude degrees eastbound form the Greenwich meridian. Coordinate systems meant for smaller areas are based on rectangular coordinate systems \[43\], the MGRS is an example of such a coordinate system.

**Projections**

A map projection is the mathematical transformation of the three dimensional spheroid into a two dimensional surface, like a computer screen or paper map. It is not possible to do this process without distorting either distance, area, shape or direction \[43\]. Hence, there are different types of map projections for different purposes.

![Map projections](image)

**Figure 3.1: Map projections \[42\]**

Map projections can be divided into three types: conic, cylindrical and planar, see Figure \[3.1\]. Every map projection has one or more lines or points of contact with the spheroid. It is only at these points and lines that the scale is true. The point of contact can either be tangent by touching the surface along one line or point, or it can be secant where the projections...
intersect with the surface and has two lines of contact. In Figure 3.1 are the conical and cylindrical projections examples of secants projections, while the planar is a tangent projection.

A map projections falls into one of the following classes: conformal (projections which preserves shape and angle), equal area projection (preserves area and size), equidistant (preserves true distance in some direction from center or along special lines), and azimuthal projection (preserves true direction from a reference point) [14].

A commonly used projection for navigation, is the Mercator projection. This projection is based on the cylindrical projection type and is tangent to equator. It preserves distance along equator but distort area and shape for locations away form the equator. Locations near the poles appears much larger than their actual size. The Web Mercator projection is a popular projection in web and mobile based applications, and is used for example in Google maps and by the OSMDroid library. EPSG:3859, EPSG:900913 and WGS 84 / Pseudo-Mercator are all identifiers for the Web Mercator projection.

**Datum**

One part of the earth mathematical model is the datum. Datum connects the spheroid with the earth surface by having one or more known locations on the earth [45]. Datums are based on a chosen spheroid, hence it will not be completely precise for the entire earth. Some datums are chosen to give the most accurate measurements in a given geographical area, known as local datums. Local datums is accurate in some parts of the world and inaccurate for others. Global datums are used to give a worldwide approximation. One example of a widely used global datum is the **World Geographic Standard (WGS)** 84 datum.

### 3.2.2 Map data types

Map data can be divided into two different categories, raster and vector maps. A vector map is a set of vector points where spatial data are represented using polygons, lines and points. Vector maps require more processing power at the client side because the vector data has to be rendered into a pixel format and styled according to a styling scheme at runtime. The strength of vector maps is that they are not degraded when zooming since the representation of different points is rendered relative to each other.
Raster maps is spatial data in a matrix format, where the smallest entity is a pixel. The huge advantage with raster maps is the low processing required when rendering the map since the data is already in a pixel format, but they are degraded when zooming. Often this is improved by using a new raster representation for each zoom level, a method known as tiling. Another difference between raster and vector maps is the storage requirement. The size of a raster map is relative to the resolution and color depth of the image, while the size of a vector map is relative to the information it contains. The size of a raster map can be significantly improved with compression.

Regardless of using vector or raster maps, the amount of data required to hold a large geographical area are usually too large to be contained within a single file or in memory on a mobile device. For this purpose online map services are used.

3.2.3 Online map services

There are several ways of serving maps from an online source including Web Map Service (WMS), Web Map Service Tile Caching (WMS-C), Web Map Tile Service (WMTS), Tile Mapping Service (TMS), XYZ-tile service and ZXY-tile service. While the first three are example of official open standards defined by Open Geospatial Consortium (OGC), the others are merely widely adopted methods (except TMS which was the predecessor of the WMTS standard).

The different types of services vary in flexibility and complexity. However, the standardized approaches seems to offer more flexible access and analysis of map data with complex queries using (mostly) web services. With the standardized approaches it is usually possible to specify the projection, which layers to include, its data type and even which styling to apply to the map. The non-standard methods, on the other hand, are usually simple and based on REST. Since they are non-standard, the implementation of the service can potentially be different for each vendor or organization, hence the API has to be manually inspected.

Bounding box and tile systems

One fundamental difference between the mentioned approaches for fetching online maps, are how they reference a geographical area. The two most common ways are either by using a bounding box or a tile system. When using a bounding box, a map is referenced using a pair of coordinates to
specify two diagonally aligned corners of the map, usually the south-east most point and the north-west most point. WMS and WMS-C both uses a bounding box.

![Diagram of WMTS Tile Matrix set](image)

Figure 3.2: WMTS Tile Matrix set

With a tile system, the earth is divided into a hierarchy of equally sized tiles (map areas) for different zoom levels, see Figure 3.2. At each zoom level the tiles are numbered using a matrix of \((x, y)\) numbers. To uniquely reference a tile the three-dimensional matrix with zoom, \(x\)-number and \(y\)-number is required, see Figure 3.3. What is important to notice is that there are multiple tile systems. For example does the tile system defined in TMS have the origin at the bottom-left corner, so the bottom-left most tile is numbered \((0,0)\), in contradictory to WMTS which uses the top-left corner as origin. Another example of a variation is that some services has an inverted zoom-level. The most common way is to have the outermost zoom level numbered 0, as in the WMTS standard.

Eventough there are different ways of referencing an area of a map, conversion between the different methods are possible. The advantage of both methods are decreased latency, CPU usage, network usage and memory consumption. It is also possible to parallel processes different part of the map by using smaller tiles than the screen size, and do pre-processing of the neighboring tiles (since they are likely to be queried next). This also enables caching at the server side.
Map sources and the Norwegian Mapping Authority

There are multiple online map source available, some requires a paid license while others are free. CAGED 1.0 uses the Free Open Source Software (FOSS) alternative OSM. The creation of OSM is a community effort, and the quality can vary a lot. It seems like maps over highly populated areas are better than maps over rural areas. Google maps is another alternative, but since Google does not allow storing map tiles off-line, it cannot be used in the CAGED application.

The Norwegian Mapping Authority is the nationwide geodata coordinator in Norway. They offers a promising WMTS service, however the service is limited to 10,000 tiles per 24 hours per IP address. This may seem like a lot, but it will not work in CAGED because it uses VPN for security. With VPN all the clients will appear with the same IP, namely the VPN public IP. In other words, with 100 users coming from the same IP the service is limited to 100 tiles per user. 100 tiles is about a square of 800x800 meters for the inner most zoom level.

Norway Digital is a community for collecting and sharing geodata in Norway. It is coordinated by the Norwegian Mapping Authority, and the Norwegian Defense is a member of the community. By being a member of the community it gives access to the highest quality map data available in Norway. The Norwegian Military Geographic Service (FMGT) is the professional authority for geograpical data in the defence, and is the
point of contact towards the Norwegian Mapping Authority.

### 3.2.4 Map libraries for Android

By having an established earth mathematical model, computing distance and referencing a point on the earth is possible. Furthermore, by mapping these locations to a pixel location, it is possible to display the map on a computer or mobile device and placing a geotagged object at the correct location on the map. There are several libraries available to simplify the work when implementing a map based application. Table 3.1 list some popular alternatives. The approach described in the official Android tutorials is by using the Google Maps Android API \[49\]. However, due to the license it cannot be used, see Chapter [1.4.1](#).

Table 3.1: Android Map libraries

<table>
<thead>
<tr>
<th>Free</th>
<th>OS</th>
<th>Offline</th>
<th>Map sources</th>
<th>Map types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Map</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Various</td>
</tr>
<tr>
<td>OSMdroid</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Various</td>
</tr>
<tr>
<td>MapsForge</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Various</td>
</tr>
<tr>
<td>Nutiteq</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Various</td>
</tr>
<tr>
<td>ArcGis runtime</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>ArcGis services</td>
</tr>
<tr>
<td>MapBox</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>WMS</td>
</tr>
</tbody>
</table>

CAGED 1.0 uses the OSMdroid Library which is, together with the MapsForge library, a FOSS alternative. Both of them seems to be mature and have an active community with frequent updates, however the OSMdroid library seems to be a better choice of various reasons. First of all is OSMdroid more flexible regarding what map sources it supports. It implements the standard TMS and support standard raster (.jpg and .png) images. MapsForge on the other hand only supports the proprietary .map vector file format and is bound to using OSM maps. By adding the extension library OSMdroid bonuspack, it is also easier to do more advanced data visualization with the OSMdroid. The MapsForge library only support simple markers and clusters. Another advantage with the OSMdroid library is that it is design to be as similar as possible to the Google maps API hence it should be easy to use for developers familiar with the Google Maps library.

Of the commercial alternatives seems ArcGis as the best alternative. The

---

\[1\] The Google Map Android library requires a premium plan if the app is used for non-private invite only use cases or if it is used for assets tracking \[50\]. The CAGED app falls into both of these categories.
reason why Nutiteq and MapBox fall through is that the developer is bound
to using their cloud services. What is important to notice, the ArcGis
runtime library is design to be used with an ArcGis server. However,
with the ArcGis server comes several interesting add-ons, like the Military
overlay. The military overlay makes it possible to add military symbols
(not limited to units/installations but also tactical graphics) based on the
APP-6 standard (see Section 3.3.2 for a definition of APP-6). The drawback
with the ArcGis library compared to OSM Droid is the license cost.

3.3 Data visualization

This section will first look at the Material Design guidelines and the
Joint Military Symbology, both are important to be familiar with when
designing an Android based military application. Then methods for
visualizing geotagged data is included as well as how this is supported
in the OSM Droid library.

3.3.1 Material Design guidelines

Android has adopted the Material Design guidelines to ensure a consistent
and coherent design across applications. Material design is a design
language launched by Google in 2014 and serves as a guide for good design.

Material is a metaphor for physical ink and paper and the goal is to
provide the user with visual cues grounded in reality. The guidelines are
continuously evolving, but as of January 2017 they involved the following
categories:

Motion is concerned with how to apply animations to an applications.
This includes constraints on how material can move, duration of
animations, a set of types of motion and how to create multiple
motions working together using what they call choreography.

Style is concerned with topics like colors, icons, typography, imagery and
writing.

Layout describes how to add color, space, scale etc to create meaning and
focus. It also includes the concept of density independent pixels (dp)
which describes how to convert pixels to ensure User Interface (UI)
elements are display uniformly across screen sizes and resolutions.

2It is possible to develop private apps with Nutiteq by signing up for their enterprise
plan at a price of 15,000 USD a year.
**Components** contains a dictionary for UI elements, describing their purpose, how they should work and how to use them. Android has built-in support for several of these components, others are provided through support libraries and a few has to be developed.

**Patterns** is concerned with user interactions and how provide feedback and display application status. It includes a definition of different types of touch gestures, their behavior and how to use them.

**Growth and communications** contains best practices on how to help users understand what they can do with the application.

**Usability** is not concerned with usability as defined in this thesis, but rather accessibility – how to ensure that the application can be used by users with various disabilities.

**Platform** describes best practices on various platforms, including iOS, Android and Web.

### 3.3.2 Joint Military Symbology

The APP-6 is the NATO standard for graphically representing military units, installations, activity or other item of military interest [36]. The standard provides a language and cultural independent system to visually represent SA information, and is often found in a NATO nations SA application.

The APP-6 standard uses different colors and shapes to differentiate between hostile, friendly, unknown and neutral entities and to identify whether it has an air, space, land or naval based unit, see Figure 3.4 on the next page. Within this shape, an icon with modifiers is used to identify the capability of the entity. Outside the shape, amplifiers are used to specify size and identity, see Figure 3.5 on the next page.

The APP-6 standard provides a rich and visually good language for representing SA related information. However, the standard is quite large and requires training to fully use and understand it.
Figure 3.4: APP-6 standard identities [36] (modified)

Figure 3.5: Different components of a APP-6 symbol [36]
3.3.3 Visualization techniques for geotagged data

Markers are often used when visualizing geotagget information and is basically an icon placed at the location of the geotagget data. Markers are often clickable and can reveal additional information when clicked. To avoid information overload when having a lot of markers in one area clustering is often used.

Clustering

Clustering is a technique adopted from data mining and is the process of grouping objects together based on similarity. To do this one or more attributes has to be chosen so it is possible to compute the distance between the objects. If the distance is within a threshold the two objects are clustered together. The distance in this context does not have to be the physical distance in meters between objects, but can be any quantitative attribute which characterize the object. For geotagged data the pixel location of the object is usually used.

OSMDroid Bonuspack clustering

The OSMDroid bonuspack has a RadiusMarkerCluster class for doing clustering. There are different types of clustering algorithms but the RadiusMarkerCluster works as follows: All markers are added to a list, the first marker is removed and put into a new cluster. Then it iterates through all the remaining markers and check the distance between them and the first marker in the new cluster. If the distance is within a defined threshold, the marker is added to the cluster and removed from the all-markers-list. This process is continued until the list is empty. When drawing the cluster onto the map, the icon is placed at the location of the first marker. This would not necessarily produce the best representation of the cluster, see Figure 3.6.

Figure 3.6a show five markers with color coding to indicate the result of the clustering algorithm. Since marker 1 is inspected first, marker 3 and 5 is put inside this cluster even though they are closer to 2 and 4. Figure 3.6b show the result of the cluster algorithm with the placing of the icon and a marker count indication. The advantage with this technique is that is fast with a complexity of Big-O = \( n^2 \).
3.3.4 Related work

In research, there are several initiatives which have looked at how to improve the situational awareness with data visualization techniques.

Livingston et al. [53] presented the Battlefield Augmented Reality System – a head mounted AR system for enhanced situational awareness for dismounted military units.

B. Liu and J. Liu [54] investigated how to solve overlapping symbols in military situational awareness with symbol fusion. They suggested two different methods, namely position adjustments (moving the position of the objects so they are not overlapping) and incorporation (merging two objects into one). The latter approach is similar to standard clustering, but in addition the resulting marker which is used as the cluster, give information about the containing clusters. For example if there are three squads in an area, the resulting cluster marker show one platoon.

Nusinov and Holsopple [55] displayed the health of friendly assets by using a bar to the right of the icon. The shortcomings with this method is that information about the asset health has to be available.

Wright and Kapler [56] presents "Bolobolgy", a method where a circle around the asset is used to visualize the range of the asset weapon, and the thickness of the circle line indicated the strength of the asset. They showed that this technique ...enhance perception and pattern recognition allowing the commander to estimate the properties of a situation more quickly and with less training then traditional methods. [56]. The shortcomings of the methods is that it requires up-to-date information about an object weapon and health.

Boukhelifa and Duke [57] investigated how to visualize uncertainty using color or blur (this study is not concerned with visualizing military assets, but service plans like pipelines). They found that color was superior over blurring, but both techniques had issues. With blurring the image was
perceived as having low quality rather than enhancing the visualization.

3.4 Information Architecture

3.4.1 Military standards

APP-6 and MGRS were identified as relevant for the CAGED application (ref Chapter 2). Since the APP-6 standard is already covered in 3.3.2, this section will only contain some basic theory about the MGRS standard.

MGRS is a coordinate system used to reference a point on the earth, see 3.2 for an explanation of a map coordinate system. MGRS is, except for around the poles, based on the Universal Transverse Mercator (UTM) coordinate system. Around the poles (south of 80 degrees south and north of 84 degrees north) it is based on the Universal Polar Stereographic (UPS). The UTM grid divides the earth into 60 zones numbered from 1 to 60 (each being 6 degrees wide) and 20 latitude bands identified by a letter (each being 8 degree high). A location within one UTM zone is given in meters easting from the zone belt and northing from the latitude band. With MGRS each of these zones are divided into smaller grids of 100 km. These squares are identified by a two letter code. The MGRS location 32VPN 45993 400 is a location in 32nd zone belt, the V latitude band and the 100 km square PN. The location is the 10 meter grid 4599 meters easting and 3400 meters northbound from 100 km square origin.

3.5 Location tracking

The built-in methods used to determine the user location with smart technology includes GPS, WiFi and Cell tower ID. The techniques are based on wave propagation theory and a known location of the source signal. The user location can be determined using triangulation based on the delay and properties of the received signal [58]. This means that the receiver has to at least have contact with three sources being spread around the user in order to determine the latitude and longitude. To include the altitude, an additional source is needed.

The properties of each of the sources vary in terms of accuracy, speed, power consumption and how they are effected by environmental conditions. This makes continuous location tracking a challenge. In most outdoor cases GPS is the most accurate source, and can give an accuracy down to a couple of meters. Since GPS depends upon line-of-sight to the satellites, it does not work for indoor location tracking and in these cases
Wireless Fidelity (WiFi) is often the most reliable source. Another drawback with GPS is that it is very power hungry, and can drain the battery in a couple of hours. Cell tower ID is the least accurate source, but it is the fastest and least power hungry source.

Which source to trust when is a trade-off between speed, accuracy and power consumption, and this is known as a location strategy in the Android documentation.

3.5.1 Location strategy and related terminology

![Location strategy model](image)

Figure 3.7: Location strategy model based on [59]

Figure 3.7 shows a model of a location strategy. The arrow represents time and the blue dots when a new location update is received. Since the accuracy of a location update will vary, a decision algorithm is used to determine if the new location is better than the old one. If it is not, the new location is discarded as illustrated with a red cross. The algorithm usually consider time and a given accuracy estimate when deciding whether to keep the location or not. The given accuracy is usually computed based on the power of the signal and the number of fixed sources.

The time from an application is starting to listen for location updates until it stops is known as the listening window. To listen for location updates usually involves activating a radio and using CPU cycles, hence consuming power. The update frequency is the number of updates per time unit. By increasing the update frequency, the convergence speed will be reduced but at the cost of increased power consumption.

Tweaking the listening window, update frequency, decision algorithm and the location source is what the location strategy is all about and is at the end a trade-off between speed, power consumption and accuracy. The challenges boils down to how to determine when to disable or enable a source (reduce the source listening window), how to determine the user state (stationary, walking, driving), and how to choose the best source and its update frequency to match the required accuracy and speed.
In research several approaches have been proposed. The approaches includes somewhat novel method as using embedded sensors to do pedestrian dead reckoning, machine learning to determine user activity and human mobility patterns to predict user movement. Section 3.5.2 gives an overview of this related work.

Pedestrian dead reckoning

Dead reckoning is an approach where location is determined without using an external source like GPS or WiFi and is achieved by calculating the user’s location based on a known starting point and the user’s speed, elapsed time and direction. In smart phones this is usually achieved using the embedded accelerometer as a pedometer and the magnetometer as a compass.

3.5.2 Related work

Research on how to optimize the location strategy can be divided into hybrid approaches, sensor management approaches and advanced approaches. The hybrid approaches combines different sources (like GPS and network) to increase the accuracy. In sensor management approaches, different sources are dynamically enabled and disabled by using embedded sensors. While advanced approaches refers to research which proposes user movement prediction using for example dead reckoning or human mobility pattern to increase the accuracy and/or reduce the power.

Hybrid approaches

60–63 are all hybrid approaches where several sources are used to determine the user’s location, including WiFi, GPS and Cell. The difference between them is how they determine when to enable or disable a source, except 62 which always has WiFi and GPS enabled to improve accuracy.

Zirari et al. 60 used the number of satellites to determine when to use WiFi to improve the accuracy. If it was less than four satellites available, WiFi was used in conjunction with GPS. This approach does not improve the battery consumption, since it requires the GPS to be constantly enabled, but improves the accuracy.

Micro-blog 61 is an approach which rely on using power hungry localization infrequently and less-power hungry approaches frequently. Micro-blog only enabled the GPS when the user had moved between multiple
WiFi access points. This approach reduces the battery consumption at the cost of reduced accuracy, but within the acceptable range for their application which were context based content sharing.

Hooi et al. [63] propose the Adaptive Signal Thresholding (AST) algorithm, which uses information about the signal strength relationship to determine whether the user is inside a building and when to enable/disable one or more signal sources. The algorithm enabled them to keep the accuracy at an acceptable value while reducing the power.

[64] is not concerned with tracking the user location specifically, but how to reduce the communication with the server keeping the information about the user’s movements. Their idea was that only the turning point of a track is of interest to others, hence they only saved the track when they detected changes in direction. If the user was moving in a straight line they did not inform the server, but when the user took a turn, the location was uploaded. This reduced the power of their application, but the track was not real-time.

Sensor management approaches

EnTracked [65, 66] proposed three sensing strategies which were combined. A movement-aware strategy used the accelerometer to decide whether the user was in a stationary state or not and disabled the GPS when this was the case. If the user was moving faster than walking speed the movement-aware strategy was disabled. They argued that a user would most likely walk before going into stationary state, and it was a waste of battery to listen for stationary states when driving. This also reduced the false positives when for example a car had stopped at a red light. The second strategy, the distance aware strategy, was a function to decide the update frequency of the GPS based on the user speed, the accepted accuracy and the GPS uncertainty. This function basically computed the traveled distance between GPS updates to be the accepted accuracy. The last strategy, the heading-aware strategy, used the embedded magnetometer to decide if the user was moving in a straight line or not. If so they disabled the GPS until the user made a turn or a threshold time was reached. The threshold time was used to reduce uncertainty with drift and change in speed.

SenseLess [67] measured the power consumption of different sensors on a Nokia N95 device when they were continuously triggered. This information was used to determine the least power intensive sensor in various situations. They found that the accelerometer consumed less
power than the GPS and WiFi so they used the accelerometer to detect when the user was stationary and disabled GPS and WiFi in these cases. This approach saved more than 58% of the energy compared to a pure hybrid approach.

SenseLoc [68] is similar to SenseLess and uses Accelerometer to determine active or inactive periods and GPS is enable and disable based on this information.

**Advanced user movement prediction approaches**

Wand et al. [69] use what they call a person’s mobility map, the users historical and common visited locations, to reduce the use of power intensive location sensors and to improve the accuracy.

Enhanced Localization Solution (ELS) [70] combines standard techniques (GPS/WiFi/Cell) with human mobility patterns, machine learning and dead reckoning to improve the accuracy and battery consumption. Dead reckoning is realized using the gyroscope, magnetometer and accelerometer. Their approach require a learning phase where the power consumption gain is not any better than using standard techniques.

**3.5.3 Location tracking with Android**

Android offers location information through the Location Manager, which handles all location information across applications. The location manager groups the location information into two types of providers the GPS provider and the Network provider (the latter wraps WiFi and Cell ID). An application can request location updates by registering a location listener to one or both of the providers and by adding a criteria which determine the update frequency. The location manager will then send a Location object to the listener whenever there is new information available, but not more often than the criteria.

The criteria can be determined by either one or a combination of the parameters time and distance the user has moved since last location update. To get the same behavior as the distance-aware strategy of EnTracked [65], the distance parameters can be used.

The attributes of a location object is shown in Figure 3.8 and information about the provider, the users location and the accuracy is relevant for the decision algorithm in the location strategy. Notice that the given accuracy is in meters with a 68% confidence.
Location relevant sensors

As research already have shown, can embedded sensors help improve the accuracy and reduce the power consumption for location tracking. Which sensors is available on an Android device and the accuracy depends on the manufacturer and model. Most Android devices have an accelerometer and a magnetometer, however the developer has to check the existence of the sensor before using it.

Sensor data is in Android accessed through the Sensor Manager, similar to the Location Manager, and handles all sensor data request across applications. The available sensors can be divided into hardware and software sensors. Hardware sensor data is data from real hardware, while software sensor data is fused or computed information from one or more hardware sensors. One example of a software sensor is the significant motion sensor which uses the accelerometer and / or gyroscope to detect when the user is moving. An event is triggered when motion is detected.

Sensor relevant for geolocation:

- Magnetic field sensor
- Orientation
- Accelerometer and Linear acceleration
- Gyroscope
- Gravity
- Step counter
Sensor relevant for detecting user activity and the environment (inside / outside a building):

- Significant motion sensor
- Ambient temperature
- Accelerometer
- Liner acceleration
Chapter 4

CAGED_2.0 Design

Multiple changes have been done to the CAGED_1.0 application in order to improve the usability. The changes are primary the solution of the medium and high priority requirements stated in Chapter 2.5.5.

When adding new features or changing existing ones, a lot of emphasis has been put on code re-usability. Effort has also been put into modularity and preserving the Entity boundary controller (EBC) – the original design pattern of CAGED_1.0.

The rest of this chapter is organized by the four categories stated in Chapter 2.5: Visualization representation, Information architecture, Interactions and Performance. Every chapter will first give an overview of what has been done within the context of the respective category, before giving details about the design choices.

4.1 Visualization representation

Table 4.1 shows an overview of the visualization improvements and what requirements they primarily solve. The design choices for each of these improvements are covered in this section.

Table 4.1: Visual representation improvements

<table>
<thead>
<tr>
<th>Change</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configurable background map from Geodata</td>
<td>2.1, 2.2</td>
</tr>
<tr>
<td>APP-6 inspired shapes</td>
<td>2.3</td>
</tr>
<tr>
<td>Clustering</td>
<td>2.4</td>
</tr>
</tbody>
</table>

EBC is a variation of the more familiar Model view controller (MVC) pattern, where the boundary is the interface to the user. This interface does not have to be a view, as in MVC, but can also be an API. The boundary interacts with controllers which can perform specific tasks and give access to the entity classes.
4.1.1 Configurable background map from Geodata

There were two issues detected related to the map; 2.1 map fidelity (the details of the map were not good enough) and 2.2 map provider (the map had to be pre-downloaded). While the first one was solved by getting the map data from a different tile source, the second issue was solved on the client side using a tile provider chain.

New tile source

A tile provider is responsible for fetching and rendering the correct tile from the configured tile source in OSMDroid. The default tile source is called "Mapnik" and fetches the map from OSM and uses a ZXY-format. The OSMDroid built-in tile sources include, Google maps, Open street map and Bing maps, but none of these sources are suitable for the CAGED application due to map quality or terms of service. As mentioned in the previous chapter, has the defence access to high quality geodata and service through the Norway digital community. With this in mind, there are two possible approaches:

1. Setting up an in-house mapping service with data from the community
2. Using a third party service offered through the Norway digital community.

The first approach gives full-control over the service and more flexibility. The drawback is that it requires competency and resources for configuration and management as well as additional software and hardware resources. With the second approach, there is little to no need for the extra resources and competency. However it does not give the same control and flexibility.

Since the FFI expressed that the preferred solution was an in-house map service, the process of getting access to the data was started. However, we did not succeed to get this data in time for this thesis. The solution was to design the CAGED software in a way that would make it simple to swap to an in-house map server at a later time. The implementation consists of three main components; MapLayerFactory, MapLayerController and MapLayerMenuDialog. The MapLayerFactory is a factory for different tile sources and map layers, and is where the MapLayerController fetches the maps. To get the solution to work with an in-house map server at a
later time, the developer simply has to change the server endpoint in the MapLayerFactory to the in-house IP.

![Map menu dialog](image)

**Figure 4.1: Map menu dialog**

As data source was the third-party REST service offered by Geodata Online used [71]. This service can serve raster background maps and satellite imagery in the Web Mercator projection (which is required by OSMDroid). To offer the flexibility for the user to choose which type of background map to use, the MapLayerController allow the user to swap between the various tile sources at runtime. The MapLayerController automatically generates a dialog menu based on the added tile sources, see Figure 4.1. By having the map layer controller as a one-stop-shopping for extending the map functionality, it is easy to add or remove map sources. An additional tile source can be added with the following line of code:

```java
mapLayerCtrl.addTileSource(ITileSource src, boolean enabled);
```

The support for adding map layers on top of the background map was also implemented. These layers can be toggled on and off and are listed under the "Kartlag" (Map layers) category in the map menu. Layers which are added to the controller has to implement the IMapLayer interface. Through the IMapLayer interface, the developer can configure the layer menu name and the menu weight. The weight will decide the order of the layers in the menu dialog. Layers with low weight are put above layers with higher numbers. It is not possible to decide the drawing order of layers (whether layer A should be drawn on top of layer B). This limitation is inherited from the the OSM Droid library to enable parallel processing of layers and avoid dependencies between layers. Layers are added to the application using the following line of code:

```java
mapLayerCtrl.addLayer(IMapLayer src, boolean enabled);
```
Client side tile provider chain

The OSMPlayer supports multiple map sources, which are called tile providers. These include a local cache, locally stored maps and online sources. In CAGED v1.0, the application was dependent on a map being pre-loaded on the soldiers’ device. This led to problems in situations where the user had to operate in unforeseen areas and did not have access to a map. This can be solved by fetching a map from an online provider, but relying solely on an online tile provider is not sufficient since the application should work in areas with a disrupted network. The solution is to use a tile provider chain [72].

A tile provider chain in the OSMPlayer library is a chain of tile providers where the system sequentially checks the availability of a tile. If the tile is not available via the first provider, the system request the next provider. The most common way to implement tile provider chains is by having one cache for fast access to the latest used tiles, one local store for areas which have to be available offline, and an online tile source for access to all other tiles.

The new implementation has added a tile provider chain using the following tile sources:

1. Local cache
2. Offline map in the <download-dir>/map directory
3. Online source

The default offline map directory is /sdcard/osmdroid/. This location can be difficult to find on some devices. To avoid that the user may not find the directory, OSMPlayer has been configured to search in the <download-dir>/map folder for offline maps. If the folder does not exists when CAGED is launched, the application will generate the directory.

Offline tile source format

OSMPlayer supports multiple off-line file formats [72], however the OSMPlayer zip format and the sqlite format are recommended [72]. The OSMPlayer zip format is basically a regular zip file with a specific hierarchy as shown in Figure 4.2. The advantage with this hierarchy is that one application can have multiple map files and multiple map sources without mixing them up. To identify the correct tile source, the folder inside the zip named <tile-source-name> in Figure 4.2 is used. Since this is inside
the zip folder and the zip folder name does not matter, multiple map files per tile source can be added. The disadvantage with the OSMDroid zip format is that the application has to inspect all the zip files when launching the application to find the correct tile source files. Hence, the off-line map directory has to be chosen wisely to avoid slow startup. This is the reason for configuring OSMDroid to inspect the `<download-dir>/map` instead of just the `<download-dir>` which potentially can hold a lot of non map zip files.

### 4.1.2 APP-6 inspired shapes

![Figure 4.3: 4.3a and 4.3b show the plot of the same situation at Åndalsnes from test using the old and new method.](image)

To improve the visualization for colorblind users a second attribute, shape, has been used to visualize the type of unit. Figure 4.3 compares
the previous design with the new. While CAGED 1.0 used circles for any types of units as well as clusters, the new design is inspired by the APP-6 specification. The APP-6 standard also makes it possible to differentiate between naval, air, space and sub-sea types of observations and this has been added to the application. See Figure 4.4 for the different types of observation supported.

<table>
<thead>
<tr>
<th></th>
<th>Friendly</th>
<th>Enemy</th>
<th>Neutral</th>
<th>Unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Land</td>
<td><img src="image" alt="Land Friendly" /></td>
<td><img src="image" alt="Land Enemy" /></td>
<td><img src="image" alt="Land Neutral" /></td>
<td><img src="image" alt="Land Unknown" /></td>
</tr>
<tr>
<td>Air</td>
<td><img src="image" alt="Air Friendly" /></td>
<td><img src="image" alt="Air Enemy" /></td>
<td><img src="image" alt="Air Neutral" /></td>
<td><img src="image" alt="Air Unknown" /></td>
</tr>
<tr>
<td>Naval</td>
<td><img src="image" alt="Naval Friendly" /></td>
<td><img src="image" alt="Naval Enemy" /></td>
<td><img src="image" alt="Naval Neutral" /></td>
<td><img src="image" alt="Naval Unknown" /></td>
</tr>
<tr>
<td>Subsea</td>
<td><img src="image" alt="Subsea Friendly" /></td>
<td><img src="image" alt="Subsea Enemy" /></td>
<td><img src="image" alt="Subsea Neutral" /></td>
<td><img src="image" alt="Subsea Unknown" /></td>
</tr>
<tr>
<td>Space</td>
<td><img src="image" alt="Space Friendly" /></td>
<td><img src="image" alt="Space Enemy" /></td>
<td><img src="image" alt="Space Neutral" /></td>
<td><img src="image" alt="Space Unknown" /></td>
</tr>
</tbody>
</table>

Figure 4.4: Implemented background shapes.

In the process of implementing the shapes it was discovered that adding an icon was rather complex and made the application quite large. For example, it required the developer to create six resources for each new icon (one for each type (friendly, hostile, neutral and unknown), one to represent observations with updated information, and one for the list of icons). This got worse when adding the support for representing air, naval, subsea and land forces, which required 18 new resource files per new icon.

The reason for the large amount of resource files was the static way of defining an observation marker, with one new resource for each representation of the observation. The solution was to separate the background, which indicates the type of observation, from the icon itself. The different observation resources are now created in runtime using a LayerDrawable (from the Android core). Two layers are added to the LayerDrawable, a background and a foreground drawable. The background is defined by the type (friendly, hostile, unknown and neutral) and capability (air, naval, subsea and space) of the observation, while the foreground drawable represents the capacity of the observation, see Figure 4.5.
4.1.3 Clustering

Requirement 2.4 addresses a problem where lack of situational awareness is caused by clustering. This has been resolved by only clustering units of the same type (friendly, hostile, unknown and neutral). The drawback with this method is that it will end up with more icons in the map, but at least it is possible to distinguish between dangerous and safe areas without having to interact with the application.

Figure 4.3 shows the old and the new method. In Figure 4.3a it is not possible to tell whether the situation is dangerous or safe around the center of the map since different types of observations are clustered together. By only clustering the same types of units and displaying these with the corresponding background, as in Figure 4.3b, it is possible to identify three friendly, two neutral and two hostile units.

To generate the new clusters, the same methods as before are employed, with the class RadiusMarkerCluster found in the OSMDroid bonus pack. The performance is slightly reduced by having four versus one cluster since four icons has to be rendered on screen instead of one.

4.2 Information architecture

Table 4.2 gives an overview of the improvements done regarding the information architecture.

<table>
<thead>
<tr>
<th>Change</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu</td>
<td>2.5</td>
</tr>
<tr>
<td>Sorting observation lists</td>
<td>2.6</td>
</tr>
<tr>
<td>Adding MGRS layer and string</td>
<td>2.7, 2.8</td>
</tr>
<tr>
<td>Filtering observations</td>
<td>2.10</td>
</tr>
</tbody>
</table>
4.2.1 Menu

The main issue with the old menu was the three-level hierarchy. In the new design the menu is implemented as a dialog organized in categories. The dialog is scrollable, so if the number of menu items exceeds the screen size it is possible to scroll down to reveal more items. To make it visually easy to separate the different categories they are displayed on separate sheets with large margin and a heading with bold typeface, see Figure 4.6. The current categories in use are "Handlinger" (Actions), "Vis innhold" (Show content) and "Innstillinger" (Settings) and are organized according to importance, with the most important category at the top.

All the items have a circular button with an icon and a descriptive text. The icon is used to give a fast meaning of the action it reveals, and the text is meant to give additional information if the icon is not intuitive to the user. To avoid the fat-finger-problem, the layout which holds the button and the text is the actual clickable area, not the button itself. The layout is configured to fill the available width and each layout has an equal weight. With an equal weight each menu item occupies the same amount of space. The red color used in CAGED 1.0 to indicate critical actions, like “Slett alle data” (Delete all data), is also used in the new design to draw attention.

To give a consistent look and feel, the background and shape of the sheets is the same as for all other dialogs. The button shapes are implemented as resources and are reused for the context menu in the detailed user view.
4.2.2 Sorting the observation list

The list of observations was difficult to navigate in CAGED 1.0 since the order was random and it only contained the title of each object. CAGED 2.0 adds a timestamp to each item, and order the items in chronological order based on the observation’s last updated timestamp. The newest observation is placed at the top, see Figure 4.7. To make the observations easier to locate a zoom-to observation-location button is added to each observation’s item. When clicking on this, the map will be centered on the observation. If the user click somewhere else in the row, a detailed view of the observation opens. To reduce the fat-finger problem, extra padding between the items in the list has been added.

4.2.3 Adding MGRS layer and string

In the analysis of CAGED 1.0 two standards were identified as relevant for the application, namely the APP-6 standard and MGRS. The APP-6 has not been implemented since the requirement was considered low priority. Hence, this chapter describes the integration of MGRS and the solution to Requirement 2.7 and 2.8.

Converting latitude and longitude to MGRS

Based on Requirement 2.7, the location of an observation should be displayed using MGRS. In CAGED 1.0 it was not possible to read out the location of an observation at all – in other words, the detailed view of a user or observation did not contain any geolocation reference string. To add a string in a view is easy in Android, however since a location is stored using the latitude longitude system, a conversion to the MGRS is necessary.
The NASA World Wind is a mapping library developed by NASA\textsuperscript{2}. This library includes an API to convert locations between different coordinate systems, which has been used in the CAGED application to do the conversion between latitude longitude system and MGRS.

### MGRS map overlay

The Norwegian Mapping Authority offers a WMS which can serve MGRS and UTM coordinate layers \textsuperscript{73}, see Figure 4.8. Since WMS uses a bounding box instead of a tile numbering system, a conversion had to be done. The conversion was implemented by following an online tutorial provided by Azavea \textsuperscript{74}. Notice that the code in the tutorial has a Massachusetts Institute of Technology (MIT) license. This should not be any issues since the MIT license is a liberal license.

![MGRS 100 km grid](image1)

(a) MGRS 100 km grid

![UTM 1 km grid](image2)

(b) UTM 1 km grid

Figure 4.8: Map overlays

### 4.2.4 Filtering observations

As stated in the discussion about Requirement 2.10, filtering should be user initiated and not remove the information completely from the view. There are two types of filtering mechanisms implemented:

1. Filter by information type using layers

2. Filter by age

\textsuperscript{2}NASA World Wind at github: [https://github.com/NASAWorldWind/WorldWindJava](https://github.com/NASAWorldWind/WorldWindJava)
Filter by information type using layers

The RadiusMarkerCluster, used for clustering of observations and users, is a type of overlay in OSMDroid. By registering the five different clusters (friendly, hostile, unknown, neutral and other users) with the map layer controller described in Chapter 4.1.1, it is possible toggle on and off the respective information. This can enable the user to focus on some specific type of information, for example all the unknown observations. The implementation deviates slightly from Requirement 2.10 since it removes information completely from the map.

Observation age

When the system is used over a longer period of time, the view is getting quite messy when a lot of observations are added. The soldiers on the ground are mainly concerned with the current situation (typically the last 24 hours). To visualize that an observation is old, observations being more than 24 hours are displayed with small icons in the map, see Figure 4.9. The figure shows that it is easy to get an overview of what have happened the last 24 hours without losing history. Since the "updated" time stamp on the observation is used to decide whether the observation is old or not, it is possible to renew an observation by editing the observation or by adding a comment to it.

![之初](image.png)

(a) Setnesmoen with no age indication   (b) Setnesmoen with age indication

Figure 4.9: Figure 4.9a and 4.9b compare a situation without and with visualizing observation age.
This filter does not remove the information completely, according to Requirement 2.10, however it is not fully user initiated. On the other hand, is it possible to renew the observation by updating the information, but it will also renew the observation for all the other users’.

An extended version of the observation age filter, could be to give an observation a "liveness" attribute to decide how old the observation has to be before considering it as old. For example, the can user choose between: 1, 12, 24 hours or never for how long the observation usually is relevant, and is only filtered out when the updated time is older than the specified time. It is also possible to not have this attribute per observation, but as a user setting, and all observations are affected by this setting.

4.3 Interactions

Table 4.3 gives an overview of the interactions improvements. Each of the improvements is described in detail in this chapter.

Table 4.3: Interactions improvements

<table>
<thead>
<tr>
<th>Change</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visualizing observation age</td>
<td>2.10, 2.12</td>
</tr>
<tr>
<td>New location target</td>
<td>2.11</td>
</tr>
<tr>
<td>Direction indication on &quot;my-location&quot;</td>
<td>2.12</td>
</tr>
<tr>
<td>Notifications</td>
<td>2.13</td>
</tr>
</tbody>
</table>

4.3.1 Visualizing observation age

See Section 4.2.4

4.3.2 New location target

Figure 4.10 shows how Requirement 2.11 has been solved, by using a target view indicating the location of the observation. To register a new observation, the user has to click and hold the desired location of the map (the same procedure as in CAGED 1.0). When a long-press is registered, a target view with a context menu, as in Figure 4.10, will be displayed and centered at the clicked location. The user can pan and zoom the map until it is placed at the correct location of the observation, avoiding the fat-finger problem. To register the observation at the targeted location, the user has to click "Legg til obs" (register observation) button.
This new design adds an additional step when registering the observation. However, it is possible to register multiple observations subsequently without having to click and hold the map again because the context menu is not removed until the user clicks "Avbryt" (Cancel).

### 4.3.3 Direction Indication

To help users navigate when they are on the move, a direction arrow has been added to the user location icon, see Figure 4.11a. The direction is obtained from the location object or computed in cases where the bearing is not provided with the location object. The direction arrow is only displayed when the user is moving. This is achieved by checking if the observation has a bearing or not. The LocationController, described in Section 4.4.1, ensures that the bearing of the observation is correct.

Figure 4.11: 4.11a and 4.11b show the user icon when moving and stationary respectively.
4.3.4 Notifications

As mentioned, did CAGED 1.0 have a notification mechanism earlier in the development phase, but it was removed for the user test due to a bug. As a quick-fix, this was reimplemented and the bug was identified and removed. However, due to limited time, no definition for what relevant information is was not established. This should done in future work to implement a smarter notification mechanism which does not notify the user of every new or updated observation.

4.4 Performance

Table 4.4: Performance improvements

<table>
<thead>
<tr>
<th>Change</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power balanced location strategy</td>
<td>2.14, 2.15, 2.16</td>
</tr>
<tr>
<td>Logging and displaying battery</td>
<td>2.15</td>
</tr>
</tbody>
</table>

4.4.1 Power balanced location strategy

Figure 4.12 shows the components implementing the location strategy. Notice that the green and white components are native Android components. The responsibility of each of the implemented components are as follows:

**ActivityMonitor**: monitors the user’s activity by using the significant motion sensor and the user movement history. It informs the LocationController when the user activity changes between active and stationary states.

**LocationController**: is responsible for activating the activity monitor and configuring the GPS based on activity changes. The LocationController implements the decision algorithm and informs all the LocationConsumers when a new location is accepted. It is implemented as a singleton class to ensure that only one instance is created.

**LocationConsumer**: are entities interested in location information. They are registered with the LocationController who ensure that they are updated whenever there is a new location available. The LocationConsumers has to implement the IMyLocationConsumer interface defined in the OSMnder library.
Activity Monitor

The activity monitor is capable of detecting user active and stationary states through two different monitors, the **StationaryStateMonitor** and the **ActiveStateMonitor**. Stationary states are detected by scheduling a task every two minutes. If the user location has not changed or the speed of the locations received during the last two minutes are zero, the user is assumed to be stationary. When a stationary state is detected, the StationaryStateMonitor is disabled to save power, and the ActiveStateMonitor is enabled.

The ActiveStateMonitor is detecting events when the user is going from stationary to active state. It is implemented by listening to the significant motion sensor offered by the Android Sensor Manager. The significant motion sensor is a software sensor which uses the accelerometer and/or gyroscope to detect when a user is going from stationary to moving state. It is rated as a low power sensor and the documentation guarantees to detect movement within 10 seconds independent of activity (driving, walking, running) [75]. The sensor has to be re-activated after it has been triggered.
Location Controller

The LocationController is the main component in the location strategy and manages all the aspects of the process, a detailed flow chart is provided in Figure 4.13. By default, it listens for both GPS and network provider to ensure the best speed and accuracy possible for both indoor and outdoor locations. The LocationController also listen to user activity state changes and when a stationary state is detected, the location controller stop listens for location updates. This reduces the battery consumption significantly, without any significant degradation of accuracy or speed.

![Figure 4.13: Location strategy flow chart](image)

The LocationController request updates from the LocationManager with an update frequency of three second and 2.5 meters. The time parameter will reduce the update frequency when moving fast at the cost of speed and accuracy, while the distance parameter will reduce the update frequency when moving slowly, hence saving power. When the application goes into paused mode, the update frequency is reduced to 20 seconds and five meters. This will reduce the power consumption at the cost of convergence speed. When the application is in paused mode (in background), it is unlikely that the user is using the application for navigation and an update frequency of 20 seconds is considered to be sufficient in this case.

The LocationController improves Requirement 2.16 through the deci-
Figure 4.14: Decision algorithm

dion algorithm. Figure 4.14 shows the flow chart of the decision algorithm. It is inspired by [59] in the way that it considers accuracy measures before timeliness to decide whether to keep the location or not. The added feature is that it is able to decide if a location is probable.

![Decision Algorithm Diagram](image)

Figure 4.15: The location is considered as probable if it is within the green area. This equals the estimated displacement vector $\vec{D}_e \pm 10$ degree and plus the distance accuracy requirement.

The green area in Figure 4.15 shows when a new location is considered as probable. This equals the estimated displacement vector $\vec{D}_e \pm 10$ degrees to cope with small directional changes and uncertainty. A distance buffer is also added to cope with small increase in speed and accuracy error of the last location. The estimated displacement vector $\vec{D}_e$ is calculated using the speed, direction and elapsed time from the last registered location $L_c$.

The shortcoming with this method is that it does not account for significant changes in speed or direction.

---

3In cases with good coverage the speed and direction is provided with the location object. In cases where they are absent it is possible to calculate them based on the displacement from the previous known location.
4.4.2 Logging and displaying battery

Requirement 2.15 (The user should not be depended on returning to a facility with corded electricity in order to recharge the device) cannot be solved within software alone, since most standard smart devices would drain their battery faster than a week even though the device is running idle. This has to be solved with hardware and routines, like giving the soldiers a battery pack or the opportunity to switch to a new device when the battery is low. To help the soldiers and the command post manage the power resources (power banks, devices etc), an attribute called battery left has been added to the user location. By doing this, the battery capacity of each user is logged every time the user location is updated. This enables the command post to monitor the battery resources of each user through Metis_2.0. The battery status is also displayed in CAGED_2.0 by appending the battery percentage to the username, see Figure 4.16.

Figure 4.16: Displaying battery capacity.
Chapter 5

Evaluation

CAGED_2.0 has been tested through five different tests – one technical test and four user tests, see Table 5.1. The technical test, T0, was a quantitative test to evaluate the performance of the power balanced location strategy. Chapter 5.1 presents the execution and the results from T0. The user tests, T1 to T4, were functional tests to measure the user satisfaction. Since the development of CAGED_2.0 has followed an agile process, the results from the user tests have been used to make adjustments to the requirements and the design. Table 5.2 gives an overview of what feature has been implemented at each of the four user tests.

In the evaluation of the user experience found in Chapter 5.2, only the results from T4 are used. There are two reasons for this: (1) T4 tested the most complete implementation of CAGED_2.0. (2) The execution and user group of T4 is most similar to the CAGED_1.0 user test. A brief overview of the execution and findings of T1, T2 and T3 are included in appendices C, D and E respectively.

Table 5.1: Tests overview

<table>
<thead>
<tr>
<th>Test</th>
<th>Context</th>
<th>Users</th>
<th>Evaluation method</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>N/A</td>
<td>4-5 phones</td>
<td>Quantitative performance data</td>
<td>3-20h a 4 reps</td>
</tr>
<tr>
<td>T1</td>
<td>Course</td>
<td>30 pax Intelligence personnel</td>
<td>Hot-wash-up session</td>
<td>2 days</td>
</tr>
<tr>
<td>T2</td>
<td>Exercise</td>
<td>15 pax Cyberservices and operations</td>
<td>Questionnaire</td>
<td>8 days</td>
</tr>
<tr>
<td>T3</td>
<td>Course</td>
<td>38 pax various</td>
<td>Questionnaire, interviews</td>
<td>5 days</td>
</tr>
<tr>
<td>T4</td>
<td>Exercise</td>
<td>19 pax Area Forces 33 pax ISTY 3 pax District command</td>
<td>Questionnaire</td>
<td>7 days</td>
</tr>
</tbody>
</table>
Table 5.2: Implementation and tests
(P: Partially, X: Fully and -: Not implemented before test)

<table>
<thead>
<tr>
<th>Category</th>
<th>Change</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visualization</td>
<td>Configurable background map</td>
<td>P^1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>APP-6 inspired shapes</td>
<td>P^2</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Clustering</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Information architecture</td>
<td>MGRS layer and string</td>
<td></td>
<td>X</td>
<td>X^3</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Sorting observation lists</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Filtering observations</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interactions</td>
<td>Direction Indication</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Notifications</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>New Location Target view</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Visualizing observation age</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>Power balanced GPS strategy</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Logging and displaying battery</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

5.1 Performance

This section evaluates the performance of the power balanced location strategy. The results are based on data gathered in test T0. T0 was conducted as a quantitative test where the location strategy in CAGED_2.0 was measured on accuracy, power consumption and convergence speed and compared with the strategy in CAGED_1.0. To have a baseline for comparison, the Google Play services Fused location provider API (from now on referred to as the baseline controller) was used. This is the recommended location strategy in the Android documentation [13].

When later referring to accuracy and convergence speed the following definitions are used: Accuracy is the distance in meters from the actual position when a new location is recorded. The convergence speed is how far the user has to travel before a new location is registered. In Figure 5.1, the accuracy is high, but the convergence speed is bad because the recorded path (the red line) does not follow the actual path (black line). Convergence speed is in this evaluation measured in meters between recorded locations.

The results show that the power consumption and accuracy have improved, while the convergence speed is preserved. This section will first give an overview of the execution of the test before presenting the findings.

1 During T1 the background map was not configurable.
2 Naval, Air, Sub-sea and Space units were not implemented for T1 and T2.
3 Minor color changes
5.1.1 T0 execution

To measure the performance of the location strategy, an app was developed called the LocationComparator. The LocationComparator had three different location controllers implemented:

1. The CAGED_1.0 location controller.
2. The CAGED_2.0 location controller.
3. The Google Play services Fused Location provider API.

The reason for building a new app for T0 was to have full control over which services were running. The app also made it easy to switch between the three different controllers.

Figure 5.2 shows the user interface of the LocationComparator. When the user clicks on one of the strategy buttons, the application starts the respective location controller. Every time the active location controller detects a new location, the position with a time stamp is appended to a log file stored locally on the device. The battery power is also logged to this file, once when starting a controller, and once when stopping a controller.

The test was executed by carrying multiple devices strapped to a backpack. Four executions were conducted, see Table 5.3. Each execution varied in length and the route was changed every time. All the executions included one or more periods of being stationary. The length of the stationary period varied from two minutes to two hours. The periods in motion included both being indoor and outdoor as well as driving, walking and skiing.

Three devices were used, one for each type of controller. The reason for having multiple devices, instead of running all the location strategies on one device, was to make it possible to measure which strategy consumed power. If the strategies have been on the same device, the update frequency

---

1 The source code can be found here: https://github.com/idafroseth/LocationStrategyComparison
of one controller could affect both the battery consumption, accuracy and convergence speed of another.

All the devices used were of type LG Nexus 5 with Android 6.0.0. A factory reset was performed on all the devices before the first execution. The devices were configured as follows:

- Adaptive brightness turned off.
- Sleep after 1 minute.
- Brightness level adjusted to maximum.
Measuring power consumption

There is no convenient way to measure power consumption accurately per app in Android\(^2\). The solution was to fetch the capacity left in percentage value from the Android PowerManager, which gives the system wide power usage. The power consumed by a strategy during an execution was computed as the power left at the time the controller was stopped minus the power left when starting the controller. Since all the devices were of same type and had the same battery size, this method gives a good indication of the battery consumption. To mitigate the error of having one device with a lower performance (e.g. one device with a worn out battery) or one device which had an background task running, the devices were used with a different strategy at each execution.

Measuring accuracy

The baseline controller was used as the reference to measure accuracy. The accuracy of a location reported by CAGED\(_1.0\) or CAGED\(_2.0\) was measured as the shortest distance from this location to the path formed by the ten locations closest in time reported by the baseline controller. The reason for choosing the ten closest locations in time instead of one, was due to the following two limitations:

1. Synchronizing the time between the devices was difficult.

2. The update frequency of the baseline controller was not high enough to give a fully real-time location.

Network Time Protocol (NTP) is a method to synchronize time over a network, and can be used to synchronize the "wall" clock in Android. However, the wall clock cannot be used to measure elapsed time in Android due to the following reason stated in the Android documentation\(^76\): The wall clock can be set by the user or the phone network (see setCurrentTimeMillis(long)), so the time may jump backwards or forwards unpredictably. This clock should only be used when correspondence with real-world dates and times is important, such as in a calendar or alarm clock application. Interval or elapsed time measurements should use a different clock.

\(^2\)The PowerTutor app has been used in early tests of CAGED\(_1.0\), and is able to display power consumption per app. The app was tested for this purpose as well, but did not produce correct output. This is most likely related to the warning displayed at start up saying: Your phone type is not recognized by PowerTutor. You may continue to use PowerTutor however we may not be able to get all of the data for your phone and the power calculations may not be accurate.
The SystemClock.elapsedRealtimeNanos() is the recommended method to measure elapsed time in Android because it is guaranteed to be monotonic, include deep sleep and tick even when the CPU is in power saving mode [76]. The SystemClock.elapsedRealtimeNanos() is given in nanoseconds from last system boot. Since it is impossible to boot all three devices at the same time, a time-stamp-button was added to the LocationComparator, see Figure 5.2. When the time stamp button was clicked, the elapsedRealtimeNanos was recorded and used as the base time. All the location updates timestamps were normalized by extracting the base time. This will not give a perfect synchronization of time, since it is difficult to click three devices exactly at the same time. However, the error was mitigated by using the locations recorded five seconds before and five seconds after from the baseline controller. In other words the devices could be 5 seconds out of synchronizations without affecting the accuracy computations.

The baseline controller was configured with an update frequency of 1000 ms (a frequency of 0 is also possible, but it will drain the battery in a couple of hours). This can affect the accuracy if an update from CAGED_1.0 or 2.0 is between two of the baseline updates, as visualized in Figure 5.3. The figure shows that the location reported by CAGED_1.0 (blue icon) is in reality much closer to the path than to the prior or subsequent location from the baseline controller. To mitigate this error, the line between the points were used to compute the shortest distance.

![Figure 5.3: Measure accuracy. The black line with the circle dots are the locations from the baseline controller, while the blue location icon is from CAGED_1.0.](image)

**Data analysis tool**

To analyze the data from T0, a small JavaScript (JS) tool was written capable of visualizing the tracks on a Google Map, and computing accuracy and convergence speed. The log files from E1 to E4 were fed to this analysis tool formatted as JavaScript Object Notation (JSON) files. To compute the accuracy, the Google Maps and Google Geometry API were used.
Alternative approaches

It is possible to use simulated locations in Android, so called mock locations. Since the CAGED_2.0 uses the significant motion sensor to detect active states, simulated locations will not give the correct result.

Limitation of the method

When starting and stopping the controllers, the devices were put on a table and all the devices were started simultaneously. It is difficult to click on multiple devices exactly at the same time and this can lead to a potential source of error. For measuring power this error was found to be negligible. For computing accuracy this limitation was overcome by using the ten closest locations in time for computing accuracy (ten locations will give five seconds prior and subsequent to the recorded location). The speed is only computed per location strategy, hence this computation is not affected by the time of start up.

5.1.2 Findings

Power consumption

Figure 5.4 shows the power consumption of each strategy during each execution. As the figure shows, CAGED_2.0 consumed from 27% to 56% less power than CAGED_1.0. The figure also shows that both CAGED_1.0 and 2.0 used significantly less power than the baseline controller. This is because the baseline controller was configured with a three times higher update frequency (in order to achieve sufficient accuracy needed as baseline).

![Figure 5.4: The power consumption in percentage per location strategy.](image-url)
Table 5.4 shows the number of locations registered by the respective controller during each execution. By comparing the table with the power consumption in Figure 5.4, it is interesting to notice the large power savings despite only saving a few locations. For example, during E4 CAGED_2.0 only had seven fewer locations than CAGED_1.0, but used about 6% less power. This can be explained as follows: CAGED_1.0 only requests updates from the Android location manager when the location has moved more than 3 meters. This means that when the user is stationary and has a good GPS signal, the location manager will not send any locations to the application. However, the GPS still has to be active. CAGED_2.0, on the other hand, is able to disable the polling of GPS updates and only listen to the less power hungry significant motion sensor.

<table>
<thead>
<tr>
<th>Execution</th>
<th>CAGED_1.0</th>
<th>CAGED_2.0</th>
<th>Baseline</th>
<th>Diff. CAGED_1.0 - 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>2702</td>
<td>2662</td>
<td>33545</td>
<td>40</td>
</tr>
<tr>
<td>E2</td>
<td>1218</td>
<td>841</td>
<td>20802</td>
<td>377</td>
</tr>
<tr>
<td>E3</td>
<td>2398</td>
<td>1788</td>
<td>36864</td>
<td>610</td>
</tr>
<tr>
<td>E4</td>
<td>2379</td>
<td>2372</td>
<td>17695</td>
<td>7</td>
</tr>
</tbody>
</table>

Accuracy

Table 5.5 shows that CAGED_1.0 and 2.0 almost had the same accuracy for execution E1, E2 and E4. However, for E3 the accuracy of CAGED_2.0 was significantly better than CAGED_1.0. This is due to the fact that the E3 execution was in an urban area which will likely reduce the quality of the GPS signal. Since CAGED_2.0 has a much more advanced decision algorithm, CAGED_2.0 is able to filter out unlikely locations. The number of recorded locations from Table 5.4 confirms this.

<table>
<thead>
<tr>
<th>Execution</th>
<th>CAGED_1.0 Mean</th>
<th>Stddev</th>
<th>CAGED_2.0 Mean</th>
<th>Stddev</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>3.722</td>
<td>5.243</td>
<td>3.714</td>
<td>4.115</td>
<td>0.008</td>
</tr>
<tr>
<td>E2</td>
<td>4.197</td>
<td>4.067</td>
<td>4.064</td>
<td>4.498</td>
<td>0.133</td>
</tr>
<tr>
<td>E3</td>
<td>6.317</td>
<td>9.466</td>
<td>3.073</td>
<td>4.441</td>
<td>3.244</td>
</tr>
<tr>
<td>E4</td>
<td>1.733</td>
<td>6.831</td>
<td>1.673</td>
<td>2.760</td>
<td>0.06</td>
</tr>
</tbody>
</table>

Figure 5.5 shows how the different controllers acts when having
poor GPS signal. In CAGED_1.0 the location is jumping around, while CAGED_2.0 is more stable when compared to the baseline.

Figure 5.5: Comparing the accuracy of three location controller when moving in an area with poor GPS signal.

Convergence speed

The convergence speed is measured in traveled distance between updates. In CAGED_1.0 and 2.0 the convergence speed depends on the velocity of the user because they request the Android Location Manager to send locations at a maximum rate (update frequency) of 3 seconds. For example, when the user is moving with a velocity of 15 m/s, the best convergence speed possible is $3s \times \frac{15m}{s} = 45m$. In this case, the mean value of the distance between updates does not give a good measure for the convergence speed. A better measure is the running average.

Figure 5.6 shows the running average of the convergence speed during each of the four executions. The velocity used in this case, is the velocity given by the Android Manager which is based on information of the GPS signal. The black line in the figure indicates the optimal convergence speed ($f(v) = 3v$) when the update frequency is 3 seconds.

As the Figure 5.6 shows, did CAGED_2.0 perform as well as CAGED_1.0 for execution E1 and E4, while for E2 and E3 CAGED_2.0 performed worse than CAGED_1.0. This can be explained by the accuracy of the GPS signal and the decision algorithm in CAGED_2.0: E1 and E4 were executed in rural areas which most likely give better GPS conditions than for E2 and E3 which were conducted in urban areas (Table 5.3 T0 Execution overview). Due to the decision algorithm in CAGED_2.0, locations with poor GPS signal is disregarded.
Figure 5.6: Running average of the convergence speed
For execution E2 the devices were placed on the floor in the backseat of a car, which would give worse GPS conditions than if they were placed by the front window. This is also reflected in the running average of the convergence speed seen in Figure 5.6. Figure 5.7 shows a plot of a sequence of locations from E2. In this area CAGED_2.0 has disregarded a lot of locations and results in a slow convergence speed. Since the traveled path had a lot of turns, the location would not be probable either, recall the decision algorithm in Chapter 4.4.1. To solve this problem, the accuracy requirement in the decision algorithm can be adjusted, but this will also reduce the accuracy. Another improvement would be to use the magnetometer to detect directional changes and use it when deciding whether a location is probable. This will affect the battery consumption.

Figure 5.7: Problems with CAGED_2.0 (Green line) under poor GPS conditions. (CAGED_1.0 plotted with a red line and Baseline with a black line).

Figure 5.6 also shows that an increase in velocity will make the locations update slower. In other words, neither CAGED_1.0 or CAGED_2.0 will be good for navigation when moving fast. If the update frequency requested from the Location Manager is reduced, the convergence speed will improve. However this can have a big impact on the battery consumption (ref. the high power consumption of the baseline controller with a update frequency of 1 second.). It is also possible to implement an adaptive update frequency, where the update frequency is changed based on the user’s velocity. If such an adaptive update frequency is implemented, it is important to remember that it will require a disabling and enabling of the Location Manager, which can potentially reduce the accuracy and speed.

3During test T0 the accuracy requirement was 5 meters (constant ACCURACY_REQUIREMENT_METERS in the LocationController), but was adjusted up to 8 meters for tests T1 – T4.
5.1.3 Summary T0

CAGED_2.0 has large power savings compared to CAGED_1.0 and improvements in the accuracy when the user is walking in urban areas. However, CAGED_2.0 is sensitive when moving in locations with poor GPS signal. For the user tests (T0 to T4) this was improved by increasing the accuracy requirement in the decision algorithm from five to eight meters. The convergence speed and power consumption has not been measured after this change, so the result in this analysis may not completely align with how the final implementation preforms. It is recommended to do a deeper analysis on how the accuracy requirement affects the power, accuracy and convergence speed to find the most optimal value.

5.2 User experience

The main goal with test T4 was to measure how much the changes in CAGED_2.0 have improved the users’ perceived usability. The data gathered in T4 was compared with the data from the SMART project user test of CAGED_1.0, hereafter referred to as the CAGED_1.0 test. The overall results shows a tendency that the users are more satisfied with the CAGED_2.0 application. However, the user group and the differences are not large enough to conclude on any point.

5.2.1 Execution

Test T4 was executed by having HV soldiers using the application during a regular exercise. A total of 54 devices were used during the test, see Figure 5.8 for the distribution of the devices. Multiple Areas participated in the exercise, but due to a limited number of devices and large geographical distances between the Areas, only one Area and the ISTY had the application. The reason for choosing the ISTY was because they partially operated in the same areas and had to communicate with the Area forces. The devices were distributed down to the squad commanders.

The following steps were performed during T4:

1. Configuration and setup of the system before deployment.
2. The users were introduced to the project and the CAGED_2.0 application through a short power-point brief.
3. The users responded to the questionnaire part I - Expectations.
4. The devices were handed out (or the application was installed on Bring Your Own Device (BOYD) devices).

5. The application was used as a tool during the HV exercise which included a training period of two days and a four days scenario play.

6. The users responded to the questionnaire part II - Experiences.

7. The devices were handed in.

Figure 5.8: T4 distribution of devices (number of devices in the parenthesis).

**Technical setup**

The setup of test T4 was similar to the setup during the SMART project user test of CAGED_1.0, see Figure 5.9. The devices were from multiple vendors as shown in Table 5.6. As already mentioned in the listing above, the devices were pre-installed and pre-configured before deployment, which included the following third-party applications:
OpenVPN Connect: for encryption between the client and the server.


DrawOnPhoto: enables the user to draw directly on a photo.

Twilight: preserves the night vision by overlaying a red filter on the screen.

During test T4 we experienced issues with VPN, which disconnected occasionally. Since CAGED (for security reasons) was dependent on VPN to communicate, see Figure 5.9, a VPN failure would cause the CAGED_2.0 to lose communication with the server. The consequence was that the users were unable to login and/or did not get any updates of the situation, before performing a re-connect of VPN. The chat client, on the other hand, was configured to bypass the VPN if it failed. This was done so it was possible to perform support when it happened and because chat had Transport Layer Security (TLS).

We were not able to identify the cause of the VPN problems completely, but they seems to be related to the keepalive and timeout settings of the VPN connection.
Table 5.6: Device types used for test T4.

<table>
<thead>
<tr>
<th>Device type</th>
<th>Nr. in use</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAT s40</td>
<td>3</td>
</tr>
<tr>
<td>Huawei P8</td>
<td>1</td>
</tr>
<tr>
<td>Huawei P9</td>
<td>2</td>
</tr>
<tr>
<td>LG Nexus 5 G</td>
<td>11</td>
</tr>
<tr>
<td>Motorola Moto G</td>
<td>6</td>
</tr>
<tr>
<td>Samsung Galaxy A5</td>
<td>1</td>
</tr>
<tr>
<td>Samsung S5 neo</td>
<td>11</td>
</tr>
<tr>
<td>Samsung Galaxy S7</td>
<td>1</td>
</tr>
<tr>
<td>Sony M4 Aqua</td>
<td>5</td>
</tr>
<tr>
<td>Sony M5</td>
<td>2</td>
</tr>
<tr>
<td>Sony Z5 Compact</td>
<td>5</td>
</tr>
<tr>
<td>CAT s60 (BYOD)</td>
<td>1</td>
</tr>
<tr>
<td>Unknown (BYOD)</td>
<td>5</td>
</tr>
</tbody>
</table>

Evaluation method

The same questionnaires as used by the SMART project during the evaluation of the CAGED_1.0, were used to gather data about the users’ experience with CAGED_2.0. There were two questionnaires, one to map the users expectations (part I) conducted prior to using the application and the other (part II) to measure the experiences with the application after usage.

The questionnaires were based on established theories and variables [6]. They included both open-ended questions and closed-ended matrix questions. A seven point Likert scale was used for the matrix questions with adaptations to fit the military domain [6].

Since the SMART project evaluated the usability of the entire system, the questionnaire also includes questions about the chat client application. These questions are viewed as irrelevant when analyzing the CAGED_2.0 application, and are not included in this evaluation.

During the CAGED_1.0 user test, only soldiers from the Area command participated. The respondents from the ISTY are not included when comparing the results from T4 with the user test of CAGED_1.0, unless otherwise stated.

The weaknesses of the experiment were that the size of the user group were to small and the results were not significant enough to conclude on any point. T4 was also conducted with a different user group, at a different
time of the year (spring versus fall) and in a different part of the country. All these factors can have affected the results. The two user groups could have had different attitude toward smart technology due to cultural differences, or the communication conditions were different due to differences in the geographical landscape. T4 was conducted in the western part of the country where the landscape has steep mountains and long fjords. While the CAGED 1.0 user test was conducted on the easter part which is rather flat. To get better control of all the variables, it would have been better to test both of the applications during the same exercise where one half of the group used the CAGED 2.0 application and the other half acted as the control group using the CAGED 1.0 application.

In addition to the above shortcomings of the method, did the SMART project identified the following weaknesses of the CAGED 1.0 test and the method: The results ... are the respondents’ perceptions at one point in time, and is not a longitudinal study. Perceptions could change over time or the answers at the time of the study possibly not representing the respondent’s typical view. Furthermore, this study had only a few respondents, making the results indications rather than more broadly generalizable conclusions [6].

Despite the mentioned shortcomings of the method used, there are trends which indicates that the users are more satisfied with the CAGED 2.0 application.

5.2.2 Findings – Questionnaire part I Expectations

This section compares the users expectations and attitude towards smart technology and the concept. The data is based on questionnaire part I. The results from T4 are compared with the data from the CAGED 1.0 user test.

The results show that the participants in T4 had an average service time of 10.4 years, this is a bit higher than in the CAGED 1.0 user test where the average was seven years. There were also a higher share of people in T4 who used Android as their private phone (63% in T4 versus 45% of the CAGED 1.0 users).

When it comes to the users’ attitude towards smart technology, it seemed that the users participating in T4 were more negative than the CAGED 1.0 user test, see Figure 5.10. At the same time the two groups seem to agree that the concept is a good idea, is realistic and is a step in the right direction. The biggest difference were found on how easy and intuitive they find smart technology, with an mean difference of 0.57 and 0.62 receptively.
Since the share of Android users were much higher in the T4 test, it is interesting to see if the attitude towards smart technology is related to the type of phone they use privately. Figure 5.11 shows that this is not the case. In cases where it is a difference between Android and iPhone users in one of the tests, the opposite result is found in the other. For example, in the CAGED_1.0 user test, the Android users responded that smart technology was easier than the iPhone users, while for T4 it was the iPhone users who thought smart technology was easier.

Figure 5.10: Expectations - Attitude

Figure 5.11: Expectations - Attitude grouped what phone they use private.
The participant of the T4 test were in general less positive when it came to the expected usefulness of the concept, except on the statement "... use of CAGED is very future-oriented", see Figure 5.12. But these results can be related to what phone the respondents uses privately. Figure 5.13 shows that the Android users are less positive than the iPhone users. It also show that the Android users of T4 seems to agree more with the Android users from the CAGED 1.0 user test.

![Figure 5.12: Expectations - Perceived usefulness](image1)

![Figure 5.13: Expectations - Perceived usefulness grouped by private phone type.](image2)
Summary Expectations

The users of T4 seems to be more negative towards smart technology and the concept compared to participant in the CAGED_1.0 user test. There can be several reasons for this. For the expected usefulness, the explanation can be that there were a larger share of Android users in the T4 test. Other reasons can be that the T4 users actually have lower expectations towards smart technology, because they are from a different part of the country. The introductory power-point brief given before the questionnaire could also have affected the result.

5.2.3 Findings – Questionnaire part II Experience

This section presents the results from the questionnaire part II. The findings are grouped in user attitude and satisfaction, CAGED system quality, and Information quality. The overall results show a tendency that the user of CAGED_2.0 are in generally more satisfied with the application, thinks the system and the quality of the information is slightly better than the users of CAGED_1.0. This is the opposite of the results from part I – Expectations, where the T4 users were more negative.

User attitude and satisfaction

The questions related to the user satisfaction of the CAGED application, indicated that the users participating in T4 were more satisfied than users participating in the test of CAGED_1.0. As shown in Figure 5.14, the users of CAGED_2.0 have scored the application higher on all points except one (very dissatisfied vs. very satisfied mean up 0.16; very displeased vs. very pleased mean up 0.23; very frustrated vs. very contented mean up 0.27; very difficult vs. very easy mean up 0.51;very problematic vs. very intuitive mean up 0.06). An even larger improvement were found on the users’ attitude (very useless vs. very useful mean up 0.25; very impractical vs. very practical mean up 0.35; very bad idea vs. very good idea mean up 0.07; very difficult vs. very easy mean up 0.51;very problematic vs. very intuitive mean up 0.06). The greatest improvement was found on how easy the application was perceived. This can be due to the changes done to the information architecture, but they can also be related to the users’ familiarity with the Android platform and/or smart technology, since there were a larger share of Android users participating in test T4.
Figure 5.14: Users’ attitude and satisfaction to the CAGED_1.0 and 2.0

Figure 5.15: User attitude and satisfaction to the CAGED_2.0 application grouped by number of times the application was opened. The results includes the responds form the ISTY.

Figure 5.15 shows the user satisfaction grouped by the number of times the application was opened by the user. It seems like users which have used the application more, are more satisfied. The reasons for this can be:

- The users which did not use the application a lot, did not have the same need as the ones who did.
• The users which did not use the application had problems with the application, and they were not able to use it. This has affected their overall impression.

• The users which used the application a lot, are more familiar with smart technology.

CAGED system quality

Figure 5.16 compares the results from T4 with the results from the user test with CAGED_1.0 on the perceived system quality of CAGED and the smart phone. The most significant change was on the statement "The smart phone runs out of battery quickly". The users of CAGED_2.0 seem to slightly disagree (mean 2.93) with this statement, while the CAGED_1.0 respondents’ answered somewhere between neutral and slightly agree (mean 4.3). This can be a result of the improved battery consumption due to the new power balanced location strategy. Several users still remark that power consumption and lack of possibilities to recharge as an issue in the open-ended questions. This can be due to the requirement 2.15 which cannot be solved in software.

Figure 5.16: CAGED and Smartphone System quality (Strongly disagree (1) vs. strongly agree (7) )

Users were also asked how they experienced the system quality in general. This included a question about the quality of the map (very bad (1) vs very good(7)). The improvement from the user test of CAGED_1.0 to T4, was 1.37 points from 5.09 to 6.46. This improvement is most likely due to changing the map from OSM to Geodata.

89
One user asked for a feature to overlay a MGRS grid and another user asked for UTM overlay in the open-ended questions. These were features implemented during during test T4, and the feedback can indicate that the feature was hard to find or not good enough.

**Information quality**

Figure 5.17: CAGED Information Quality (Strongly disagree (1) vs. strongly agree (7)).

**Figure 5.17** shows the results from the questions about the information quality of the CAGED application. The most significant improvement was on the questions of the accuracy of CAGED (mean up 0.62). This can be a result of improved accuracy with the new target view when registering an observation. The accuracy of the location strategy has also improved, which can also be a reason for why users thinks that the accuracy is better in CAGED_2.0.

From **Figure 5.17** it seems like the users of CAGED_2.0 may have experienced more problems with the application than the users of CAGED_1.0 (mean up 0.18). Of the eight responders who had scored this question as four or higher, did four of them mention VPN problems in the open-ended questions, two mentioned battery as an issue and the last two answered blank.

The answers about the information quality also indicates that the users did not have as much information to accomplish their mission during T4 compared to the CAGED_1.0 user test (mean down 0.27 points). There are no changes in the application which give the users less information in
CAGED_2.0 compared to CAGED_1.0. What has changed is how the menu is organized and where the button to reveal a list over all the observations are placed. A cause of this reported difference can also be due to the way the users used the application or that they had a different information need due to different missions. During the CAGED_1.0 user test the primary communication channel (military radio) failed, so the smart phones were the only way they could communicate, and may have affected the result.
Chapter 6

Conclusion

The objective of this thesis was to improve the usability of the CAGED_1.0 application. By doing this, it contributes to show that smart technology can be a viable alternative to specialized military equipment. The objective has been reached by answering the problem statement stated in Chapter 1.

Part one of the problem statement was What are the main usability requirements of the CAGED application? The highest prioritized requirements were found to be the map base, the location strategy, the power consumption and the visualization of observations. Improvements to the menu design, user interactions and the information architecture were found to be of medium priority.

The second problem statement was: How to best solve these challenges with the known limitations of smart technology using state of the art and applying them to Android? This is answered in Chapters 3 and 4 and an overview of the uncovered requirements, whether they have been solved and what the solution was, are provided in Table 6.1 on the next page.

After implementing the medium and high priority requirements, there is a tendency that the users are more satisfied with the application and they seems to agree that the usability of the application has improved. The users agreed that the map base has improved and the power consumption is better. However, the size of the user group and the significance of the results are not high enough to draw a conclusion.

The power consumption was improved through a power balanced location strategy. This strategy is able to save power whenever the user is stationary for more than two minutes. The power savings of CAGED_2.0 has been found to be as high as 50%. The power balanced location strategy is also slightly more accurate than the previous solution, and matches the convergence speed of CAGED_1.0 in most situations. The caveat is that the CAGED_2.0 location strategy is sensitive when the user is moving in areas
Table 6.1: Requirement versus implementation status ordered by priority and requirement number (Y* in the table means that the requirement has been solved, but the solution can be improved).

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Pri</th>
<th>Solved</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Map fidelity</td>
<td>H</td>
<td>Y</td>
<td>Configurable background map from Geodata</td>
</tr>
<tr>
<td>2.2 Map provider</td>
<td>H</td>
<td>Y*</td>
<td>Configurable background map from Geodata</td>
</tr>
<tr>
<td>2.3 Observations - color and shape</td>
<td>H</td>
<td>Y</td>
<td>APP-6 inspired shapes</td>
</tr>
<tr>
<td>2.4 Limit power consumption</td>
<td>H</td>
<td>P</td>
<td>Power balanced location strategy</td>
</tr>
<tr>
<td>2.5 Power consumption (charging)</td>
<td>H</td>
<td>P</td>
<td>Power balanced location strategy and Logging and displaying battery</td>
</tr>
<tr>
<td>2.6 Stable GPS routine</td>
<td>H</td>
<td>Y*</td>
<td>Power balanced location strategy</td>
</tr>
<tr>
<td>2.7 Clustering</td>
<td>M</td>
<td>Y</td>
<td>Cluster per status (Friendly, hostile, unknown and neutral)</td>
</tr>
<tr>
<td>2.8 Menu design</td>
<td>M</td>
<td>Y</td>
<td>Menu with categories</td>
</tr>
<tr>
<td>2.9 Ordering of information</td>
<td>M</td>
<td>Y</td>
<td>Sorting observation list and adding a timestamp and a zoom-to-observation-location button.</td>
</tr>
<tr>
<td>2.10 Displaying observations with MGRS</td>
<td>M</td>
<td>Y</td>
<td>Converting Lat Long to MGRS</td>
</tr>
<tr>
<td>2.11 MGRS map overlay</td>
<td>M</td>
<td>Y</td>
<td>Adding MGRS layer from Norwegian mapping authority</td>
</tr>
<tr>
<td>2.12 Fat finger problem</td>
<td>M</td>
<td>Y</td>
<td>Location target</td>
</tr>
<tr>
<td>2.13 Time to interact</td>
<td>M</td>
<td>P</td>
<td>Visualizing observation age and adding a direction indication on &quot;my-location&quot;</td>
</tr>
<tr>
<td>2.14 Notifications</td>
<td>M</td>
<td>P</td>
<td>Re-implemented the previous notification</td>
</tr>
<tr>
<td>2.15 Military standards (APP-6)</td>
<td>L</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>2.16 Filtering</td>
<td>L</td>
<td>P</td>
<td>Visualizing observation age and Filtering observations</td>
</tr>
</tbody>
</table>

Table 6.1 shows that some of the requirements were only partially solved. The reason for only partially fulfilling the high prioritized requirements 2.14 and 2.15 is:

**Requirement 2.14 Limit power consumption:** Optimizing the power consumption can be done in many ways, like limiting the number of CPU cycles (processing required by the app), reducing the network communication by improving the communication pattern and reducing the use of power hungry sensors like the GPS. Since the goal with this thesis was to improve the usability, it was important to get the power consumption down to an acceptable level while still have time for other high and medium prioritized requirements. The reason for
choosing the location strategy as a way to improve the power consumption, was that it needed work to improve the accuracy anyway. In addition was the CAGED_1.0 application network usage reduced by the SMART project by implementing version control.

**Requirement 2.15 Power consumption (charging):** this requirement states: *The user should not be dependent on returning to a facility with cored electricity in order to recharge the device.* This cannot be solved in software alone, but has to be solved through routines, hardware (power banks) and logistics. The requirement has been partially solved by reducing the power with an improved location strategy and by logging the power status to the server so battery resources can be managed better by the logistics element.

The reason for only partially solving the medium prioritized requirements 2.12 Time to interact and 2.13 Notifications was due to the limited time frame.

All in all, the objective of this thesis was reached: Almost all the requirements were completely fulfilled, with only four requirements that were partially fulfilled. Hence, we can say that the work has met the requirements, and the goal of improving the usability of the CAGED application has been reached. The requirements being partially fulfilled should be investigated as part of future work, to further enhance the demonstrator.
Chapter 7

Future work

During the work with this thesis, several areas were discovered where there is room for further improvement. However, due to the limited timeframe of this thesis, it was not possible to pursue these ideas further. This chapter gives an overview of known open issues and suggestions for further improvements to CAGED.

7.1 In-house map server

The map was implemented using a third party online source. If full control over the mapping service is desired, an in-house tile service should be implemented. The spatial data and styling scheme can be made available by the FMGT along with a description of how to implement the service using the open source software MapServer [77]. It is recommended to use the MapCache [78] software together with the MapServer for better performance.

7.2 Power consumption

The power consumption was only improved through the geolocation mechanism. The app can also be optimized in other means, like reducing the polling frequency or the communication pattern. It is recommended that this is further investigated with a high priority.

7.3 Speed of the GPS routine

The technical test of CAGED_2.0 found that the convergence speed of the geolocation mechanism can be slow in cases with bad GPS coverage. This
can be solved by either tweaking the LocationController parameters or by implementing user movement prediction with dead reckoning.

The geolocation mechanism can also be improved for situations when moving fast. The update frequency is 3 seconds which means that when driving in 50 km/h the location is updated every 40 meters \( \left( \frac{50 \text{km}}{3600 \text{s}} \right) = 13.9 \text{m/s} \Rightarrow 13.9 \text{m/s} \times 3 \text{s} = 41 \text{m} \). If the update frequency is decreased, the convergence speed will improve but it can have a big impact on the battery consumption. It is also possible to implement an adaptive update frequency, where the update frequency is changed based on the user’s velocity. If such an adaptive update frequency is implemented, it is important to remember that it will cause a disabling and enabling of the Location Manager which can positionally reduce the accuracy and speed. In other words, the adaptive update frequency cannot change continuously but can for example change when passing one or two threshold values.

7.4 Time to interact

It is still potential when it comes to reducing the required attention by the users. It is recommended to look at mechanism offered by smart technology which can enhance this. One example can be to provide the users with a headset with a remote control, and allow the users to control the most basic functionality of the app with it. The headset can also be used to give audio feedback of new or updated information, think of it as a type of advanced notification mechanism.

7.5 Notifications

The notification service has room for improvements. In CAGED 2.0 the users are notified for every new or updated observation. This also accounts for information which is not relevant to the user. A definition of what relevant information is has to be established and can for example be the observations physical distance from the user, if the user has marked the observation as a favorite and if the user is the creator or editor of an observation with updated information. A more advanced mechanism when clicking on a notification is also recommended to implement. In CAGED 2.0 clicking on a notification opens the application. If it had also zoomed to the location of the observation which has been updated, the user would not have to look for the information.
7.6 The low prioritized requirements

7.6.1 The APP-6 standard

Requirement 2.9 has not been implemented, but it is recommended to do it. The requirement was: *It should be possible to toggle between the APP-6 standard and a more easy-to-use icon set in the CAGED application.*

7.6.2 Filtering

Since filtering was a low prioritized requirement, only some simple mechanism was implemented at the end of the development of CAGED 2.0. The first one was to enable the user to toggle on and off information through the MapLayerController. The shortcoming with it is that there is no indication in the UI that a layer is toggled off and the user can forget that he/she has done it.

The second mechanism for filtering was to make old observations visually less important by reducing their size. In CAGED 2.0 an observation is considered old when it has not been updated the last 24 hours. It is recommended to implement a mechanism where the liveness time of an observation is configurable. For example if an observation has an attribute which indicate its liveness threshold, different type of information can have different liveness. Another approach would be to have the liveness time as a user preference. This way the user can decide how old observations has to be before reducing their size.

7.6.3 Security

Security was defined as out of scope for this thesis, however problems with the VPN connection were found to affect the overall impression of the application. It is recommended that the security is investigated further. Both to identify the problems with the VPN solution and to evaluate alternative approaches such as putting the security per app on the transport layer.
Bibliography


[30] OpenStreetMap contributors. OpenStreetMap powers map data on thousands of web sites, mobile apps, and hardware devices. URL: https://www.openstreetmap.org/about (visited on 01/12/2017).


[75] Sensor types. URL: https://source.android.com/devices/sensors/sensor-types#significant_motion (visited on 05/16/2017).


[87] PostgreSQL. The PostgreSQL Global Development Group. URL: https://www.postgresql.org (visited on 05/31/2017).

Glossary

API  Application Programming Interface.
APP-6  Allied Procedural Publication 6.
AR  Augmented Reality.
AST  Adaptive Signal Thresholding.
BFT  Blue Force Tracking.
BOYD  Bring Your Own Device.
C2  Command and Control.
C2IS  Command and Control Information System.
CAGED  Communication Application with Geographical Element Data.
CD&E  Concept Development and Experimentation.
CEI  Collective Environment Interpretation.
COTS  Commercial off-the-shelf.
CTO  Cyberservices and operations.
EBC  Entity boundary controller.
ELS  Enhanced Localization Solution.
FAB  Floating Action Button.
FFI  Norwegian Defence Research Establishment.
FOSS  Free Open Source Software.
GPS  Global Positioning System.
HE  Heuristic Evaluation.
HTTP  Hypertext Transfer Protocol.
HV  Norwegian Home Guard.
ISO  International Organization for Standardization.
ISTY  Rapid-reaction Intervention Forces.
JSON  Javascript Object Notation.
mAh  miliampere hour.
MANET  Mobile ad hoc network.
MGRS  Military grid reference system.
MVC  Model view controller.
NATO  North Atlantic Treaty Organization.
NFC  Near Field Communication.
NNEC  NATO Network Enabled Capability.
NorBMS  Norwegian Battlefield Management System.
OGC  Open Geospatial Consortium.
OSM  OpenStreetMap.
PROMISE  PROject Multi-touch Information System Exchange.
REST  Representational State Transfer.
SA  Situational awareness.
TIGR  Tactical Ground Reporting System.
TMS  Tile Mapping Service.
UI  User Interface.
UPS  Universal Polar Stereographic.
USB  Universal Serial Bus.
USDoD  United States Department of Defense.
UTM   Universal Transverse Mercator.
VPN   Virtual Private Network.
VR    Virtual Reality.
WGS   World Geographic Standard.
WiFi  Wireless Fidelity.
WMS   Web Map Service.
WMS-C Web Map Service Tile Caching.
WMTS  Web Map Tile Service.
XML   eXtensible Markup Language.
Appendices
Appendix A

Android essentials

Android is an open source software stack based on the Linux kernel and the project is lead by Google [79, 80]. An overview of the software stack is provided in Figure A.1.

The Android framework provides the entire feature-set of the Android operating system through a Java API, however the way of developing an Android application is a bit different from traditional Java. In order to fully utilize the Android framework and develop a stable application, the developer has to as a minimum be familiar with the following concepts: Android components, the Manifest file, the Activity life-cycle and Application Resources.

A.1 Android components

An Android application does not have a single main method entry point, but multiple which is either an Activity, a Service, a Content provider or a Broadcast receiver [81].

An activity represents a single UI screen and is found in most Android applications. A quick overview of the different components are provided in Table A.1.

A.2 The manifest file

The manifest file is required in any Android application and has to be located in the root directory of the app [82]. The file is written in eXtensible Markup Language (XML) and informs the Android framework about various details about the application. This includes, amongst others, permissions required by the app, information about the application
Table A.1: Android components

<table>
<thead>
<tr>
<th>Component</th>
<th>What is it used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>Single screen UI. An Android application with a user interface has to have at least one class extending an Android Activity class</td>
</tr>
<tr>
<td>Services</td>
<td>Long-running operation done in a separate thread in the background. Examples are playing music while the user does something else or communicating with a server.</td>
</tr>
<tr>
<td>Content providers</td>
<td>Manage shared set of app data, like contact information.</td>
</tr>
<tr>
<td>Broadcast receivers</td>
<td>Responds to system wide announcements like screen turned off, screen orientation changed and home button clicked</td>
</tr>
</tbody>
</table>
components, how they are connected and which one to run at launch time, the minimum Android API required and which sensors are required by the application.

A.3 Application lifecycle

The user is in control of an Android device and can freely navigate between applications, close an application or put the device into sleep mode. Situations can arise which are not initiated by the user himself, like receiving a phone call or be put into sleep-mode after an inactive period. To handle these events, the Android framework defines different states and callback methods when the activity moves from one state to another and this is known as the lifecycle. Figure A.2 is from the Android documentation and shows the different states an Activity can be in and what callbacks are associated with them (note that other Android components have slightly different lifecycle and callback methods). Depending on where the application is in its lifecycle it can be a candidate for memory disposal, and to ensure proper behavior, the developer has to override the different callback methods to ensure that the state is preserved and resources are properly detached / attached.

An activity goes through the created states only once during the entire application lifecycle, when the application is first launched. The other states are as follows

**Started:** A transition state between created or stopped and resumed. The application is visible to the user, but will shortly move to the Resumed state.

**Resumed:** When in the resumed state, the application is in the foreground and has the user focus.

**Paused:** Another activity is visible on top of the application. When in paused state the activity can be killed in a situation where the memory is extremely low.

**Stopped:** The application is running in the background and is alive but is not attached to the window manager. The application will be killed if memory is needed elsewhere.
A.4 Application resources

Application resources are located within the src/main/res folder of an Android application, and includes images, audio files and the visual representation of the app. The latter includes, amongst other, layouts, menus, styles, colors and animations, and all are written in XML. This gives a loose coupling between the application functionality and its appearance, hence the look and feel can easily be changed without breaking the application logic. This also easily enables multilingual support, by having one locale file for each language.

A.5 Material Design guidelines

Android has adopted the Material Design guidelines to ensure a consistent and coherent design across applications [51]. Material design is a design language launched by Google in 2014 and serves as a guide for good design [52].

Material is a metaphor for physical ink and paper and the goal is to provide the user with visual cues grounded in reality [52]. The guidelines are continuously evolving, but as of January 2017 they involved the following categories:

Motion is concerned with how to apply animations to an applications. This includes constraints on how material can move, duration of animations, a set of types of motion and how to create multiple motions working together using what they call choreography.

Style is concerned with topics like colors, icons, typography, imagery and
writing.

**Layout** describes how to add color, space, scale etc to create meaning and focus. It also includes the concept of density independent pixels (dp) which describes how to convert pixels to ensure UI elements being displayed uniformly across screen sizes and resolutions.

**Components** contains a dictionary for UI elements, describing their purpose, how they should work and how to use them. Android has built-in support for several of these components, others are provided through support libraries and a few has to be developed.

**Patterns** is concerned with user interactions and how provide feedback and display application status. It includes a definition of different types of touch gestures, their behavior and how to use them.

**Growth and communications** contains best practices on how to help users understand what they can do with the application.

**Usability** is not concerned with usability as defined in this thesis, but rather accessibility - how to ensure that the application can be used by users with various disabilities.

**Platform** describes best practices on various platforms, including iOS, Android and Web.
Appendix B

Athena_2.0

Due to several limitation with Ahena_1.0, and the most severe were:

- There was no relations in the database which led to an inconsistent database in some cases.
- Athena_1.0 was hard to configure and set-up.
- Athena_1.0 had performance issues when hashing the password.

B.1 Tools and frameworks

Athena_2.0 is built using the Spring Framework [84], a framework for building Java-based enterprise applications. We chose to use the Spring Boot [85] which simplify configuration as well as deployment. Spring Boot come with an embedded web sever and can be run as a basic Java application. The Spring Data JPA [86] was used for JPA based data access. This removes the need for boilerplate code and simplify the data access layer. It has a built in Object/Relational Mapper and all data access and transactions can be written as Plain Old Java Objects (POJO). Spring JPA can also auto generate the database tables, based on configuration done with either XML or annotations. We chose to use annotations. PostgreSQL [87] was chosen as the database, a FOSS database which is easily integrated with Spring. Swagger was used for auto generating the REST end-point documentation. Maven was used for project management and dependency injection.
B.2 Application structure

The MVC pattern was used for Athena_2.0. The most important packages are:

no ffi ep1667 athena controller: contains all the REST controller classes which implement the REST end-points.

no ffi ep1667 athena exception: contains all the Hypertext Transfer Protocol (HTTP) exception classes are located in this package.

no ffi ep1667 athena model: The domain model of Athena 2.0 is implemented in this package using entity classes with getters and setters. The relational model is automatically generated based on the annotation found in these classes (@OneToOne, @OneToMany and @ManyToMany).

no ffi ep1667 athena repository: contains the Database Access Objects. These interface classes extends the CrudRepository class. Which is a skeleton class used by Spring Data JPA to generate basic data access. Extended data access can be achieved by using named queries, JPA Query language or SQL [86].

no ffi ep1667 athena services: contains the services which implements the business logic.

no ffi ep1667 athena security: contains security configuration.

System configuration is done in: src/main/resources/application.properties

B.3 Future work

The following is proposed as future work:

Integrate Metis_2.0: It is possible to integrate a React based application into a Spring based Java application, a tutorial is provided in [88]. By integrating Metis_2.0 with Athena_2.0 it will simplify deployment.

Transport Layer Security (TLS): With TLS it is possible with authentication using certificate. This will improve the usability of the CAGED application and at the same time it makes the system more secure. It should also be considered whether TLS should replace VPN.
B.4 Domain Model

Figure B.1 shows the domain model of Athena_2.0.

![Domain Model Diagram]

Figure B.1: Domain model
Appendix C

Summary and feedback from test T1

C.1 Summary of the activity

The CAGED application was presented and tested during an intelligence course by HV-11 at Åndalsnes during the period 13th-14th of February. The test represent the first qualitative test during this work. It was 40 participants, where 25 of these used their own device (BOYD), nine used a device belonging to the project and the rest got to try other users devices. The age of the users spanned from mid twenties to mid fifty.

The first day, the 13th of February, was used for installation. A local webserver was configured giving the users access to their VPN-profile, an offline map file and a compiled version of the CAGED application. All the users with BOYD performed the installation themselves, with guidance and assistance. This was partly a success, and about five had to get extra assistance in order to get the application working. One out of the users with BOYD were not able to install the application due to having an old Android version (he had version 4.1.1 and CAGED requires v.5.0). The installation took about one hour including a short lecture.

The second day, the 14th of February, started by briefing the users about the project and the system. Amongst the users were district commander and G2 HV-11. The functionality of Metis was also demonstrated. Then the users were free to use the app during a 30 minute test session (some of the BOYD users had already used the application since installing it the previous day). After the test session a plenary feedback session was held and all the users were encourge to express their views on the application and the system.
A lot of input were given during the feedback session, both related to CAGED and Metis. It was a common conception that the functionality of the Titans system was something the HV need. A lot of the feedbacks was on the possibility of naval awareness, since the naval HV is about to be merged with the land forces. HV-11 is also a district with a long coastline, and the naval situational awareness is extra important for them.

C.2 Feedback

The feedback is grouped into what part of the system they are related to. The feedback related to CAGED is found in Table C.1 while Metis is found in Table C.2. Feedback regarding the system as a whole is found in Table C.3 and discovered bugs in Table C.4.

<table>
<thead>
<tr>
<th>Table C.1: Test 1 feedback CAGED</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category</strong></td>
</tr>
<tr>
<td>Direction</td>
</tr>
<tr>
<td>Sector</td>
</tr>
<tr>
<td>Standard format</td>
</tr>
<tr>
<td>Observation</td>
</tr>
<tr>
<td>Alarm</td>
</tr>
<tr>
<td>Hatched to indicate age</td>
</tr>
<tr>
<td>Visualization</td>
</tr>
<tr>
<td>Icons</td>
</tr>
<tr>
<td>Air icons</td>
</tr>
<tr>
<td>‘Other’ icon</td>
</tr>
<tr>
<td>Category</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>Filter and search</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Observations</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Table C.3: Test 1 feedback System

<table>
<thead>
<tr>
<th>Category</th>
<th>Sub-category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphic</td>
<td>Exposure lines</td>
<td>Possibility to register exposure lines which indicates a line in the terrain where the soldier when crossing the line will be exposed to the enemy.</td>
</tr>
<tr>
<td>Maritim</td>
<td>Symbols and map</td>
<td>Since the naval HV is discontinued, the need to be able to create a naval situational awareness picture is increased.</td>
</tr>
<tr>
<td>Urgency</td>
<td></td>
<td>The possibility to tag an observation with degree of urgency. Examples are routine and flash. This will make it easier to prioritise.</td>
</tr>
<tr>
<td>Approved</td>
<td></td>
<td>The users wanted a system to quality assure the observation added. For example by a user at a higher level, e.g., the Metis operator can confirm that an observation is of good quality and valid. This makes it easier to detect duplicates and to remove information which is not relevant or valid.</td>
</tr>
<tr>
<td>Priority Information Request (PIR) and Critical Information Request (CIR) tagging</td>
<td></td>
<td>In the observation order are various PIR and CIR listed to give the soldiers a lead on what information is needed. If the it was possible to tag an observation with which PIR or CIR this observation belonged to, it would be easier to categorise and prioritise information.</td>
</tr>
<tr>
<td>Friendly</td>
<td></td>
<td>These users did not want the possibility to add friendly forces observation, since it is usually sensitive information. This does not account for tracking the soldiers.</td>
</tr>
<tr>
<td>Observation</td>
<td>Link two observations</td>
<td>Possibility to link to related observations.</td>
</tr>
<tr>
<td></td>
<td>&quot;Best before date&quot;</td>
<td>Possibility to add a &quot;best before date&quot;, indicating that the information is outdated when passing this date.</td>
</tr>
<tr>
<td>Integration</td>
<td>&quot;Watermark&quot;</td>
<td>If integrating Athena with other systems, it should be possible to identify which observations are registered through Athena.</td>
</tr>
</tbody>
</table>
Table C.4: Test 1 Reported bugs

<table>
<thead>
<tr>
<th>Category</th>
<th>Sub-category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>location</td>
<td>Since CAGED uses VPN the users location will be placed at Kjeller when using Network location.</td>
</tr>
<tr>
<td>Position</td>
<td>Mock locations</td>
<td>It is possible to make apps which manipulate the user location, known as Mock location. These locations are meant for testing purposes, but there can be background tasks which the user is not aware of manipulating the user location. To avoid this, it is possible to do a check in the decision algorithm wether the location comes from a mock provider.</td>
</tr>
<tr>
<td>Offline</td>
<td>Add observation</td>
<td>If the user is not connected with the server and add an observation, the observation would not appear in the map (the user registering the observation map) before the user is online.</td>
</tr>
</tbody>
</table>
Appendix D

Summary and feedback from test T2

D.1 Summary of the activity

T2, second functional test, was an initiative from the Norwegian Armed Forces Cyberdefence who needed a system to perform command and control of the Cyberservices and operations (CTO) during the NATO exercise Winter Fusion in Finnmark. CTO’s primary task is to ensure that the maneuver units in the area of operations has a secure data connection. Their way of operating differs slightly form the intended usage, however the opportunity was used for quality assure the system and to gather user feedback through questionnaires.

There were restriction on using cell phones in the area of operation, so only six users could use the system during the exercise. The functionality which were implemented for this test were a new location strategy, a map from Geodata as background map, new menu, new cluster mechanism and a direction indication.

D.2 Findings

During the test it was discovered bugs with the location tracking mechanism which could cause it to hang. There were also problems with the VPN connection. However, the results show that the users were satisfied with the application with a mean slightly above 5, and they had an general positive attitude towards the system, see table D.1. From Table D.2 it seems like the CTO have a rather different information need since they answered that they disagree on the question CAGED "...gave me sufficient
information to accomplish the mission.”. This was also reflected in the open ended-questions where they asked for new features which have not been reported by to many HV soldiers. This included:

- Professional weather forecast.

- Technical documentation (needed to maintain the equipment in service).

- Material logistics.

On the system quality, see Table D.3 they tend to agree that the quality was good, except for the network coverage.

Table D.1: CAGED Attitude and user satisfaction

<table>
<thead>
<tr>
<th>Comparison</th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very dissatisfied (1) vs. very satisfied (7)</td>
<td>5.0</td>
<td>0.70</td>
</tr>
<tr>
<td>Very displeased (1) vs. very pleased (7)</td>
<td>5.0</td>
<td>0.70</td>
</tr>
<tr>
<td>Very frustrated (1) vs. very contented (7)</td>
<td>5.2</td>
<td>0.83</td>
</tr>
<tr>
<td>Very terrible (1) vs. very delighted (7)</td>
<td>5.2</td>
<td>0.83</td>
</tr>
<tr>
<td>Very useless (1) vs. very useful (7)</td>
<td>5.2</td>
<td>1.09</td>
</tr>
<tr>
<td>Very impractical (1) vs. very practical (7)</td>
<td>5.2</td>
<td>0.83</td>
</tr>
<tr>
<td>Very bad idea (1) vs. very good idea (7)</td>
<td>5.4</td>
<td>1.14</td>
</tr>
<tr>
<td>Very difficult (1) vs. very easy (7)</td>
<td>5.2</td>
<td>0.83</td>
</tr>
<tr>
<td>Very problematic (1) vs. very intuitive (7)</td>
<td>5.0</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Table D.2: CAGED information quality (Strongly disagree (1) vs. strongly agree (7) )

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>...gave me the information I needed.</td>
<td>3.8</td>
<td>0.44</td>
</tr>
<tr>
<td>...gave me the exact result I expected.</td>
<td>3.8</td>
<td>0.83</td>
</tr>
<tr>
<td>...gave me sufficient information to accomplish the mission.</td>
<td>2</td>
<td>1.23</td>
</tr>
<tr>
<td>I had to make workarounds because of error in the software.</td>
<td>3.8</td>
<td>1.78</td>
</tr>
<tr>
<td>I am satisfied with the accuracy (of CAGED).</td>
<td>4.0</td>
<td>0.90</td>
</tr>
<tr>
<td>...gives me useful results for solving a mission.</td>
<td>3.8</td>
<td>1.48</td>
</tr>
<tr>
<td>...gives me useful information for solving questions and problems</td>
<td>3.6</td>
<td>1.67</td>
</tr>
</tbody>
</table>

Table D.3: CAGED system quality

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The smartphone runs out of battery quickly.</td>
<td>1.8</td>
<td>0.83</td>
</tr>
<tr>
<td>I used the map (in CAGED) very often.</td>
<td>4.6</td>
<td>0.89</td>
</tr>
<tr>
<td>I often checked CAGED for new information.</td>
<td>3.8</td>
<td>1.92</td>
</tr>
<tr>
<td>I often reported observations via CAGED.</td>
<td>2.6</td>
<td>1.34</td>
</tr>
<tr>
<td>It was useful to see the others’ positions.</td>
<td>5.4</td>
<td>0.55</td>
</tr>
<tr>
<td>The observations in CAGED were relevant (for me).</td>
<td>4.2</td>
<td>0.84</td>
</tr>
</tbody>
</table>
Table D.4: Smartphone and System quality
(Very bad (1) vs. very good (7) and do not know (8))

<table>
<thead>
<tr>
<th>CAGED</th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network (coverage)</td>
<td>3.5</td>
<td>0.58</td>
</tr>
<tr>
<td>Map</td>
<td>5.5</td>
<td>0.58</td>
</tr>
<tr>
<td>Time it took to upload map</td>
<td>5.25</td>
<td>1.71</td>
</tr>
<tr>
<td>Register observations</td>
<td>6.0</td>
<td>0.82</td>
</tr>
<tr>
<td>View others’ observations</td>
<td>4.5</td>
<td>1.91</td>
</tr>
<tr>
<td>Registering own position</td>
<td>5.24</td>
<td>2.22</td>
</tr>
<tr>
<td>View others’ position</td>
<td>4.5</td>
<td>1.73</td>
</tr>
<tr>
<td>Starting the CAGED-app</td>
<td>4.75</td>
<td>1.25</td>
</tr>
<tr>
<td>Getting started with CAGED</td>
<td>5.25</td>
<td>0.5</td>
</tr>
<tr>
<td>First impression of CAGED</td>
<td>5.5</td>
<td>0.57</td>
</tr>
</tbody>
</table>
Appendix E

Summary and feedback from test T3

E.1 Summary of the activity

The third functional test, T3, was with the HV school at Dombås during a squad leader course. During these courses they train multiple scenarios in cycles of 24 hours where they operate for about 12-16 hours and evaluate their performance at the end of the day. The same scenario or a new scenario is played the next day where the roles are rotated among the squad members. The devices were collected after each scenario. During this test, two different versions of the app were used, one version with the new location strategy and the other with the old strategy. Data about the power consumption was gathered by the server, and user feedback was gathered using questionnaires, interviews and observations.

E.2 Findings

E.2.1 Interviews

Two users were interview after the T3 test. The interviews were in Norwegian, so only a summary of the interviews are included here.

They both think the application is good, and is easy to use. They both said that it was very useful to see the other users on the map. The application has an operational value. The weakness of the system is that is does not always have updated information due to the VPN problem. This makes them lose trust to the system.

They think it is very intuitive to use the application, and as a tool it seems very useful, especially for team members. The Squad and Platoon
leader does not have time to use the application.

On improvement of the application did they say:

- It should be easier to view images, and not need to click the x button between every image.

- In some cases was the location inaccurate, this should be fixed.

- It would have been very nice to have a feature where it was possible to measure distance. This is something we have to do on every mission to place our fire power.

- It should have been possible to get the operational order via the application

- It would have been nice to get a MGRS grid of my position.

**E.2.2 Questionnaire**

Table E.1: T3 CAGED Attitude and user satisfaction

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very dissatisfied (1) vs. very satisfied (7)</td>
<td>5.25</td>
<td>1.12</td>
</tr>
<tr>
<td>Very displeased (1) vs. very pleased (7)</td>
<td>5.20</td>
<td>1.15</td>
</tr>
<tr>
<td>Very frustrated (1) vs. very contented (7)</td>
<td>5.38</td>
<td>1.36</td>
</tr>
<tr>
<td>Very terrible (1) vs. very delighted (7)</td>
<td>4.92</td>
<td>1.40</td>
</tr>
<tr>
<td>Very useless (1) vs. very useful (7)</td>
<td>5.11</td>
<td>1.39</td>
</tr>
<tr>
<td>Very impractical (1) vs. very practical (7)</td>
<td>4.90</td>
<td>1.37</td>
</tr>
<tr>
<td>Very bad idea (1) vs. very good idea (7)</td>
<td>5.61</td>
<td>1.21</td>
</tr>
<tr>
<td>Very difficult (1) vs. very easy (7)</td>
<td>5.56</td>
<td>1.19</td>
</tr>
<tr>
<td>Very problematic (1) vs. very intuitive (7)</td>
<td>5.28</td>
<td>1.43</td>
</tr>
</tbody>
</table>
Table E.2: T3 CAGED information quality (Strongly disagree (1) vs. strongly agree (7) )

<table>
<thead>
<tr>
<th>Statement</th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>...gave me the information I needed.</td>
<td>4.39</td>
<td>1.57</td>
</tr>
<tr>
<td>...gave me the exact result I expected.</td>
<td>4.37</td>
<td>1.44</td>
</tr>
<tr>
<td>...gave me sufficient information to accomplish the mission.</td>
<td>3.94</td>
<td>1.66</td>
</tr>
<tr>
<td>I had to make workarounds because of error in the software.</td>
<td>3.15</td>
<td>2.04</td>
</tr>
<tr>
<td>I am satisfied with the accuracy (of CAGED).</td>
<td>4.46</td>
<td>1.42</td>
</tr>
<tr>
<td>...gives me useful results for solving a mission.</td>
<td>4.94</td>
<td>1.51</td>
</tr>
<tr>
<td>...gives me useful information for solving questions and problems</td>
<td>4.61</td>
<td>1.55</td>
</tr>
</tbody>
</table>

Table E.3: T3 CAGED system quality

<table>
<thead>
<tr>
<th>Statement</th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>The smartphone runs out of battery quickly.</td>
<td>1.8</td>
<td>1.53</td>
</tr>
<tr>
<td>I used the map (in CAGED) very often.</td>
<td>3.79</td>
<td>2.13</td>
</tr>
<tr>
<td>I often checked CAGED for new information.</td>
<td>2.65</td>
<td>1.76</td>
</tr>
<tr>
<td>I often reported observations via CAGED.</td>
<td>2.30</td>
<td>1.70</td>
</tr>
<tr>
<td>It was useful to see the others’ positions.</td>
<td>5.21</td>
<td>1.72</td>
</tr>
<tr>
<td>The observations in CAGED were relevant (for me).</td>
<td>4.02</td>
<td>1.71</td>
</tr>
</tbody>
</table>
Table E.4: T3 Smartphone and System quality
(Very bad (1) vs. very good (7) and do not know (8))

<table>
<thead>
<tr>
<th>CAGED</th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network (coverage)</td>
<td>5.47</td>
<td>1.68</td>
</tr>
<tr>
<td>Map</td>
<td>5.97</td>
<td>1.32</td>
</tr>
<tr>
<td>Time it took to upload map</td>
<td>5.78</td>
<td>1.51</td>
</tr>
<tr>
<td>Register observations</td>
<td>5.82</td>
<td>1.56</td>
</tr>
<tr>
<td>View others’ observations</td>
<td>5.55</td>
<td>1.74</td>
</tr>
<tr>
<td>Registering own position</td>
<td>5.76</td>
<td>1.50</td>
</tr>
<tr>
<td>View others’ position</td>
<td>5.26</td>
<td>1.48</td>
</tr>
<tr>
<td>Starting the CAGED-app</td>
<td>4.33</td>
<td>2.16</td>
</tr>
<tr>
<td>Getting started with CAGED</td>
<td>4.94</td>
<td>1.89</td>
</tr>
<tr>
<td>First impression of CAGED</td>
<td>4.76</td>
<td>1.54</td>
</tr>
</tbody>
</table>
Appendix F

Nielsens 10 heuristics

This appendix contains Nilsens 10 heuristics and are an exact copy of the listing found in [29].

F.1 Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

F.2 Match between system and the real world

The system should speak the users’ language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

F.3 User control and freedom

Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

F.4 Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
F.5  Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action. (Read full article on preventing user errors.)

F.6  Recognition rather than recall

Minimize the user’s memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate. (Read full article on recognition vs. recall in UX.)

F.7  Flexibility and efficiency of use

Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

F.8  Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

F.9  Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

F.10  Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such
information should be easy to search, focused on the user’s task, list concrete steps to be carried out, and not be too large.
Appendix G

CAGED_1.0 Heuristic evaluation

The HE is based on Nielsens ten heuristics with the additional interpretation provided by Kuparinen et.al \cite{89}. The chapter is organized with one section for each heuristic.

G.1 Visibility of system status

Feedback

The app does not provide proper feedback when new information is available in the app, like a new observation is added or updated. This is really troublesome when someone uses the comment field of the observation to request new information about the observation.

G.2 Match between the system and the real world

Map fidelity

The map lack a lot of information about obvious physical information, especially in rural areas. For example when zooming into Olasvevegen 135, Lillehammer with zoom level 16, no buildings are displayed, with the oldest building within the view was built around 1910.

User location is not stable

The user location is jumping around and seems like changing between the GPS location and cell-location. This issue seems to be more prominent when the user is located inside a building.
APP-6 standard

The system does not follow APP-6 conventions when displaying SA information. For trained personnel this can be confusing. This is a design choice, since APP-6 require time to learn, and may not be suitable for intentional end-user of the CAGED app.

G.3 User control and freedom

Lack of return options

When viewing details about an observation it is not possible to exit back to the map without using the return button. The same accounts when the second menu level is exploded. Since this information is displayed using dialog it is not added to the back-stack and when using the back button it returns to the map instead of the last action.

Updating location of an observation

It is not possible to update the location of an observation. Since the precision of the touch screen is not too good, it is situation where the location is a bit off, and there is a need to update this location.

G.4 Consistency and standards

Main menu icon

The main menu icon is a cog icon, which is mainly used for settings menu. This can prevent the user from finding relevant actions since it is not perceived as a menu button, see Figure G.1.

Extensive use of FAB

There is an extensive use of FAB. According the Material design guidelines: "Only one floating action button is recommended per screen to represent the most
Figure G.2: Using the same icons for different types of observations can be a problem for color blind users.

common action

Menu does not follow platform conventions

The platform conventions for navigations says: *Focus attention on important destinations by displaying them in tabs or in the side navigation, and de-emphasize inessential content by displaying it in less prominent locations.*

Hence the main menu items should be located in a tabbed menu and not as a FAB.

NATO Standards

Does not follow the APP-6 standard. An issue related to this is that all the different observations are displayed using a circle icon, including clustered observations, which can be a problem for users with color blindness. For example would it be difficult to determine if the observation is hostile (RED) or neutral (GREEN), see Figure G.2.

Unfortunate clustering

Different types of observations are clustered together, and makes it difficult to spot if it is a dangerous or safe area. The clustering is also too aggressive since it cluster down to the largest zoom level and when two or more observations are closer than 100 pixels. This implies that the user has to

---


click the cluster to reveal information about the observations within the cluster, see Figure G.3.

G.5 Error Prevention

The system has adequate error prevention by using warning messages and by storing data locally when the network is not available.

G.6 Recognition rather than recall

When the user are viewing an observation selected from the list of all observations it is not possible to return to the list. Instead the user must go through two level of menus before he/she is back to the list. In the meantime the user may have forgotten which observation he/she viewed last.

G.7 Flexibility and efficient use

Hierarchical menus

Hierarchical menu makes the user interface inefficient. The menu has three levels and the choice of items in level 1 vs level 2 is not too obvious. See Figure G.4.
Finding the location of an observation

When viewing an observation detail its location is not displayed only the distance from the current users. This is troublesome when the user access this information from a list of observations. There should at least be possible to get the distance and direction or the observation physical location.

G.8 Aesthetic and minimalistic design

The application is in general very good at this, and no issues are found.

G.9 Help users recognize, diagnose and recover from errors

The application is good at preventing errors both due to user interactions and when the network or GPS it not available.

G.10 Help and Documentation

There is no documentation built into the app. When using the app the first time it is not obvious that a new observation is added with a long click on the map, and with no documentation this is difficult to discover.
Appendix H

Results

H.1 Performance

Table H.1: Measured power consumption – location strategy comparison

<table>
<thead>
<tr>
<th>Execution</th>
<th>CAGED_1.0</th>
<th>CAGED_2.0</th>
<th>Google baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Start</td>
<td>End</td>
<td>Used</td>
</tr>
<tr>
<td>E1</td>
<td>96%</td>
<td>67%</td>
<td>29%</td>
</tr>
<tr>
<td>E2</td>
<td>99%</td>
<td>74%</td>
<td>25%</td>
</tr>
<tr>
<td>E3</td>
<td>100%</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>E4</td>
<td>99%</td>
<td>78%</td>
<td>21%</td>
</tr>
</tbody>
</table>

Table H.2: T0 Accuracy

<table>
<thead>
<tr>
<th>Execution</th>
<th>CAGED_1.0</th>
<th>CAGED_2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Median</td>
</tr>
<tr>
<td>E1</td>
<td>3.287</td>
<td>2.451</td>
</tr>
<tr>
<td>E2</td>
<td>3.897</td>
<td>2.996</td>
</tr>
<tr>
<td>E3</td>
<td>4.809</td>
<td>2.291</td>
</tr>
<tr>
<td>E4</td>
<td>1.697</td>
<td>1.043</td>
</tr>
</tbody>
</table>
### H.2 Expectations

#### Table H.3: Attitude

<table>
<thead>
<tr>
<th></th>
<th>CAGED_1.0</th>
<th></th>
<th>CAGED_2.0</th>
<th></th>
<th>Diff (2.0-1.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>STD</td>
<td>Mean</td>
<td>STD</td>
<td></td>
</tr>
<tr>
<td>Very useless (1) vs. very useful (7)</td>
<td>6.76</td>
<td>0.502</td>
<td>6.56</td>
<td>0.629</td>
<td>-0.20</td>
</tr>
<tr>
<td>Very impractical (1) vs. very practical (7)</td>
<td>6.70</td>
<td>0.529</td>
<td>6.31</td>
<td>0.704</td>
<td>-0.38</td>
</tr>
<tr>
<td>Very difficult (1) vs. very easy (7)</td>
<td>6.30</td>
<td>0.810</td>
<td>5.73</td>
<td>0.799</td>
<td>-0.57</td>
</tr>
<tr>
<td>Very problematic (1) vs. very intuitive (7)</td>
<td>6.42</td>
<td>0.751</td>
<td>5.80</td>
<td>0.862</td>
<td>-0.62</td>
</tr>
<tr>
<td>Very bad idea (1) vs. very good idea (7)</td>
<td>6.42</td>
<td>0.830</td>
<td>6.31</td>
<td>1.078</td>
<td>-0.11</td>
</tr>
<tr>
<td>Very unrealistic (1) vs. very realistic (7)</td>
<td>5.91</td>
<td>1.100</td>
<td>6.00</td>
<td>1.195</td>
<td>0.09</td>
</tr>
<tr>
<td>A development in the wrong direction (1) vs. a development in the right direction (7)</td>
<td>6.61</td>
<td>0.704</td>
<td>6.53</td>
<td>0.915</td>
<td>-0.06</td>
</tr>
</tbody>
</table>

#### Table H.4: Attitude by private phone type.

<table>
<thead>
<tr>
<th></th>
<th>CAGED_1.0</th>
<th></th>
<th>CAGED_2.0</th>
<th></th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Android</td>
<td>IPhone</td>
<td>Diff</td>
<td>Android</td>
<td>IPhone</td>
</tr>
<tr>
<td>Very useless (1) vs. very useful</td>
<td>6.80</td>
<td>6.72</td>
<td>0.08</td>
<td>6.5</td>
<td>6.67</td>
</tr>
<tr>
<td>Very impractical (1) vs. very practical</td>
<td>6.67</td>
<td>6.72</td>
<td>-0.06</td>
<td>6.3</td>
<td>6.33</td>
</tr>
<tr>
<td>Very difficult (1) vs. very easy</td>
<td>6.40</td>
<td>6.22</td>
<td>0.18</td>
<td>5.67</td>
<td>5.83</td>
</tr>
<tr>
<td>Very problematic (1) vs. very intuitive</td>
<td>6.47</td>
<td>6.39</td>
<td>0.08</td>
<td>5.78</td>
<td>5.83</td>
</tr>
<tr>
<td>Very bad idea (1) vs. very good idea</td>
<td>6.40</td>
<td>6.44</td>
<td>-0.04</td>
<td>6.4</td>
<td>6.17</td>
</tr>
<tr>
<td>Very unrealistic (1) vs. very realistic</td>
<td>5.60</td>
<td>6.17</td>
<td>-0.57</td>
<td>6.11</td>
<td>5.83</td>
</tr>
<tr>
<td>A development in the wrong direction (1) vs. a development in the right direction</td>
<td>6.60</td>
<td>6.61</td>
<td>-0.01</td>
<td>6.67</td>
<td>6.33</td>
</tr>
</tbody>
</table>
### Table H.5: Perceived usefulness

<table>
<thead>
<tr>
<th></th>
<th>CAGED_1.0</th>
<th>CAGED_2.0</th>
<th>Diff (2.0-1.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I would find mobile information platforms useful in my job.</td>
<td>6.39</td>
<td>6.00</td>
<td>-0.39</td>
</tr>
<tr>
<td>Using mobile information platforms enables me to accomplish tasks more quickly.</td>
<td>5.88</td>
<td>5.56</td>
<td>-0.32</td>
</tr>
<tr>
<td>Using mobile information platforms increases my productivity.</td>
<td>5.91</td>
<td>5.47</td>
<td>-0.44</td>
</tr>
<tr>
<td>... use of CAGED is very future-oriented.</td>
<td>5.82</td>
<td>6.69</td>
<td>0.87</td>
</tr>
<tr>
<td>... use of CAGED (or similar apps) is necessary.</td>
<td>5.67</td>
<td>5.19</td>
<td>-0.48</td>
</tr>
<tr>
<td>... use of CAGED (or similar apps) is unavoidable.</td>
<td>5.79</td>
<td>5.63</td>
<td>-0.16</td>
</tr>
<tr>
<td>... increase the effectiveness of the mission.</td>
<td>5.97</td>
<td>6.00</td>
<td>0.03</td>
</tr>
<tr>
<td>... spend less time on routine tasks.</td>
<td>5.45</td>
<td>5.38</td>
<td>-0.08</td>
</tr>
<tr>
<td>... improve the quality of results.</td>
<td>5.85</td>
<td>6.00</td>
<td>0.15</td>
</tr>
</tbody>
</table>
H.3 Experience

Table H.6: CAGED Attitude and user satisfaction

<table>
<thead>
<tr>
<th></th>
<th>CAGED_1.0 Mean</th>
<th>CAGED_2.0 Mean</th>
<th>Diff (2.0-1.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very dissatisfied (1)</td>
<td>5.38</td>
<td>5.54</td>
<td>0.16</td>
</tr>
<tr>
<td>vs. very satisfied (7)</td>
<td>1.185</td>
<td>0.78</td>
<td></td>
</tr>
<tr>
<td>Very displeased (1)</td>
<td>5.34</td>
<td>5.57</td>
<td>0.23</td>
</tr>
<tr>
<td>vs. very pleased (7)</td>
<td>1.315</td>
<td>0.85</td>
<td></td>
</tr>
<tr>
<td>Very frustrated (1)</td>
<td>5.34</td>
<td>5.62</td>
<td>0.27</td>
</tr>
<tr>
<td>vs. very contented (7)</td>
<td>1.359</td>
<td>1.12</td>
<td></td>
</tr>
<tr>
<td>Very terrible (1)</td>
<td>5.50</td>
<td>5.38</td>
<td>-0.12</td>
</tr>
<tr>
<td>vs. very delighted (7)</td>
<td>1.218</td>
<td>1.12</td>
<td></td>
</tr>
<tr>
<td>Very useless (1)</td>
<td>6.00</td>
<td>6.23</td>
<td>0.23</td>
</tr>
<tr>
<td>vs. very useful (7)</td>
<td>1.107</td>
<td>0.93</td>
<td></td>
</tr>
<tr>
<td>Very impractical (1)</td>
<td>5.50</td>
<td>5.85</td>
<td>0.35</td>
</tr>
<tr>
<td>vs. very practical (7)</td>
<td>1.391</td>
<td>0.99</td>
<td></td>
</tr>
<tr>
<td>Very bad idea (1)</td>
<td>6.47</td>
<td>6.54</td>
<td>0.07</td>
</tr>
<tr>
<td>vs. very good idea (7)</td>
<td>0.879</td>
<td>0.52</td>
<td></td>
</tr>
<tr>
<td>Very difficult (1)</td>
<td>5.72</td>
<td>6.23</td>
<td>0.51</td>
</tr>
<tr>
<td>vs. very easy (7)</td>
<td>1.224</td>
<td>0.73</td>
<td></td>
</tr>
<tr>
<td>Very problematic (1)</td>
<td>5.78</td>
<td>5.85</td>
<td>0.06</td>
</tr>
<tr>
<td>vs. very intuitive (7)</td>
<td>1.157</td>
<td>0.90</td>
<td></td>
</tr>
</tbody>
</table>
Table H.7: CAGED Information Quality  
(Strongly disagree (1) vs. strongly agree (7) )

<table>
<thead>
<tr>
<th>CAGED_1.0</th>
<th>CAGED_2.0</th>
<th>Diff (2.0-1.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>STD</td>
<td>Mean</td>
</tr>
<tr>
<td>...gave me the information I needed.</td>
<td>5.13</td>
<td>1.129</td>
</tr>
<tr>
<td>...gave me the exact result I expected.</td>
<td>4.66</td>
<td>1.234</td>
</tr>
<tr>
<td>...gave me sufficient information to accomplish the mission.</td>
<td>5.13</td>
<td>1.264</td>
</tr>
<tr>
<td>I had to make workarounds because of error in the software.</td>
<td>2.97</td>
<td>1.892</td>
</tr>
<tr>
<td>I am satisfied with the accuracy (of CAGED).</td>
<td>4.53</td>
<td>1.502</td>
</tr>
<tr>
<td>...gives me useful results for solving a mission.</td>
<td>5.44</td>
<td>1.151</td>
</tr>
<tr>
<td>...gives me useful information for solving questions and problems</td>
<td>5.16</td>
<td>1.440</td>
</tr>
</tbody>
</table>

Table H.8: CAGED and Smartphone System quality  
(Strongly disagree (1) vs. strongly agree (7) )

<table>
<thead>
<tr>
<th>CAGED_1.0</th>
<th>CAGED_2.0</th>
<th>Diff (2.0-1.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>STD</td>
<td>Mean</td>
</tr>
<tr>
<td>The smartphone runs out of battery quickly.</td>
<td>4.30</td>
<td>2.218</td>
</tr>
<tr>
<td>I used the map (in CAGED) very often.</td>
<td>5.45</td>
<td>1.563</td>
</tr>
<tr>
<td>I often checked CAGED for new information.</td>
<td>5.48</td>
<td>1.564</td>
</tr>
<tr>
<td>I often reported observations via CAGED.</td>
<td>3.39</td>
<td>1.853</td>
</tr>
<tr>
<td>It was useful to see the others' positions.</td>
<td>6.21</td>
<td>1.219</td>
</tr>
<tr>
<td>The observations in CAGED were relevant (for me).</td>
<td>5.58</td>
<td>1.251</td>
</tr>
</tbody>
</table>
Table H.9: System quality
(Very bad (1) vs. very good (7) and do not know (8))

<table>
<thead>
<tr>
<th></th>
<th>CAGED_1.0</th>
<th>CAGED_2.0</th>
<th>Diff (2.0-1.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>N(8)</td>
<td>Mean</td>
</tr>
<tr>
<td>Network (coverage)</td>
<td>6.00</td>
<td>0</td>
<td>6.21</td>
</tr>
<tr>
<td>Map</td>
<td>5.09</td>
<td>0</td>
<td>6.46</td>
</tr>
<tr>
<td>Time it took to upload map</td>
<td>5.36</td>
<td>0</td>
<td>6.0</td>
</tr>
<tr>
<td>Register observations</td>
<td>5.81</td>
<td>1</td>
<td>5.79</td>
</tr>
<tr>
<td>View others’ observations</td>
<td>5.82</td>
<td>0</td>
<td>6.71</td>
</tr>
<tr>
<td>Registering own position</td>
<td>5.85</td>
<td>6</td>
<td>6.43</td>
</tr>
<tr>
<td>View others’ position</td>
<td>5.58</td>
<td>0</td>
<td>6.36</td>
</tr>
<tr>
<td>Starting the CAGED-app</td>
<td>5.58</td>
<td>0</td>
<td>4.93</td>
</tr>
<tr>
<td>Getting started with CAGED</td>
<td>6.06</td>
<td>0</td>
<td>5.57</td>
</tr>
<tr>
<td>First impression of CAGED</td>
<td>5.91</td>
<td>0</td>
<td>6.50</td>
</tr>
</tbody>
</table>

Table H.10: CAGED Attitude and user satisfaction versus times they opened CAGED (using three equally sized groups of 11 users)

<table>
<thead>
<tr>
<th></th>
<th>0 - 80 times</th>
<th>81 - 187</th>
<th>&gt; 188</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>STD</td>
<td>Mean</td>
</tr>
<tr>
<td>Very dissatisfied (1) vs. very satisfied (7)</td>
<td>5.00</td>
<td>1.13</td>
<td>5.62</td>
</tr>
<tr>
<td>Very displeased (1) vs. very pleased (7)</td>
<td>4.83</td>
<td>1.27</td>
<td>5.46</td>
</tr>
<tr>
<td>Very frustrated (1) vs. very contented (7)</td>
<td>5.17</td>
<td>1.27</td>
<td>6.23</td>
</tr>
<tr>
<td>Very terrible (1) vs. very delighted (7)</td>
<td>4.75</td>
<td>0.97</td>
<td>5.77</td>
</tr>
<tr>
<td>Very useless (1) vs. very useful (7)</td>
<td>5.25</td>
<td>1.06</td>
<td>6.00</td>
</tr>
<tr>
<td>Very impractical (1) vs. very practical (7)</td>
<td>4.83</td>
<td>1.19</td>
<td>6.08</td>
</tr>
<tr>
<td>Very bad idea (1) vs. very good idea (7)</td>
<td>5.75</td>
<td>1.06</td>
<td>6.54</td>
</tr>
<tr>
<td>Very difficult (1) vs. very easy (7)</td>
<td>5.67</td>
<td>0.98</td>
<td>6.31</td>
</tr>
<tr>
<td>Very problematic (1) vs. very intuitive (7)</td>
<td>5.33</td>
<td>1.07</td>
<td>6.15</td>
</tr>
</tbody>
</table>
Appendix I

Technical guide CAGED

This guide describes how to install and start developing for the CAGED application. The guide will first give some preq

I.1 Prerequisites

The CAGED is an Android application built using Android studio and Git for version control. The guide will provide the reader with the minimum of the commands needed to get started developing, but it assume the following:

- Basic knowledge of Git (if not, the official documentation is a good starting point)
- Installed and working version of git
- Installed and working version of Android studio with Android SDK 5.0
- Access to the Gitlab repository at [FFI]

I.2 CAGED versions and Git branches

Git has been used for version control when developing CAGED and three categories of branches has been used:

1. The main branch
2. Feature branches
3. Production branches
The main branch

The main branch represent the latest stable work. Only a few core peoples have write access to this branch, and the rest has to perform a merge request in order to save changes to this branch. In Gitlab this is users with Master access to the repository.

Feature branches

A feature branch represent a new feature or a large change in the code and uses the following naming convention: feature/MK-<serial_number>-<branch_name>. When and if the new feature has been found stable it has been merge into the main branch, notice that a feature does not have to been merged with the master branch and can contain "broken" code (e.g. experimental changes).

NOTICE: By the time of this writing the feature branch integrating CAGED with Athena 2.0 back-end has not been merge with master, but is named feature/MK-200-NEW-ATHENA.

Production branches

Production branches are used to keep the history of the different versions used during the different tests. These branches are called production/W<week_number>-<test_location>. For some of the tests, changes were made during the test and the respective production branch represent how the code was at the end of the test.

I.3 Getting started

The following steps has to be performed to make changes to CAGED:

1. Download the code
2. Import the project in Android Studio
3. Change the code
4. Push the code to the remote repo

Downloading the code

The first step is to download the code from Gitlab.
#Move to a directory where you like to save the code
```bash
cd Users/<username>/Documents/workspace/
```

#Clone the remote repo (requires VPN)
```bash
git clone http://<username>@<git_ip>/titans/caged.git
```

#A new folder with the name caged will be created, move to this folder
```bash
cd caged
```

#List all local and remote branches
```bash
git branch -a
```

#For every remote branch you like to have locally, run:
```bash
git branch --track <branch_name>
```

#Or track all the remote branches with:
```bash
for i in $(git branch -r |grep -vE "HEAD|master"); do
git branch --track ${i#/*} $i; done
```

**Importing the project into Android Studio**

1. Start Android Studio
2. Click "Open an existing Android Studio project" in the main window
3. Locate and click the caged folder downloaded in the previous step

**Push the code to the remote repository**

```bash
#Create a new branch
git checkout -b
"feature/MK<next_number>--<descriptive-name>"

#Add the branch as a remote branch
git push --set-upstream origin
"feature/MK<next_number>--<descriptive-name>"
```