Information supply during military missions. Relevant or not?

Final Report BSc Project IN3405

June 24, 2011

Bram Beernink
Arjen Goedegebure
Niels van Kaam
Remco van der Zon

Bachelor committee

TNO Soesterberg  Delft University of Technology, faculty EWI
Ir. Marc Grootjen  Prof.dr. M.A. Neerincx
Ir. Wouter Vos  Drs. P.R. van Nieuwenhuizen
Summary

During modern military operations the environment is dynamic and unpredictable. There are many actors in the area of the operation, military as well as non-military, friendly and non-friendly. Combined with personnel reductions this creates the need for a flexible and multifunctional military organisation.

In an organisation where less people need to do more it is very important to utilise the skills of every individual. Utilising those skills will create a “power to the edge” structured organisation where decision making is also done at lower levels. This results in groups in the field becoming more autonomous. For autonomous groups to be able to make decisions there is a need to facilitate anticipation and increase efficiency. To reach these goals it is important that a group has a good situational awareness.

Providing a person with not only information directly addressed to him, but also with information that was sent to others, could increase situational awareness. Autonomous groups could then be better able to anticipate on future events and make better decisions. However, to prevent information overload a relevance system determines if a message is also relevant by calculating a relevance score for every user. The main research question is how the process of transmitting information can be automated and how this can facilitate anticipation and increase efficiency.

This question was researched as a bachelor project for Computer Science at TU Delft. The project was part of the Smart Operations programme at TNO. The execution of the bachelor project was carried out using the situated Cognitive Engineering (sCE) methodology. For a large part this methodology has determined how the project was executed in different phases.

In the implemented system, each user can use an application to view the messages that are relevant for him. Latent Semantic Indexing was used for the determination of relevance based on message content, while a fuzzy expert system was used for the overall determination of relevance, which was also based on message metadata.

A pilot was conducted with a specialist of the Royal Netherlands Navy. A dynamic scenario was simulated and the system determined the relevance for all messages in the scenario and showed the expert a message list ordered by the relevance. The expert indicated that the developed system has the potential to assist in facilitating anticipation and increase efficiency. He also provided recommendations to improve the current system.

The recommendation by the expert was to distinguish separate different message types, for example situational reports and planning messages. Next to this recommendation of the expert we also came to our own recommendations. The first recommendation is to obtain a better dataset for the pilot. The current dataset contained few messages from the environment other than the command center. Furthermore some system components to simulate a scenario are not implemented yet. To do a better test it would be useful to implement a Mission Task Manager, Role Manager and Position Emulator.
Preface

This report is the Final Report of our BSc project for our bachelor Computer Science at the TU Delft. During our Bsc project we worked for TNO Soesterberg to deliver a prototype which gives messages a relevance score to military operators during a military mission. From the start of this project, we have enjoyed the execution of this project. We are proud of the produced results.

We would like to thank our TNO supervisors Ir. Marc Grootjen and Ir. Wouter Vos for their continuous assistance at all times. We thank our TU Delft supervisor Prof.dr. M.A. Neerincx for his supervision and insights. We also would like to thank Maarten van Zomeren for his assistance to familiarize us with Trex and RDF. Not in the least we thank Wytse Posthumus for his always quick responses in adding and improving a LaTeX export features to the situated Cognitive Engineering Tool, which we intensively used.
# Table of Contents

1 Introduction .......................................................... 1

2 Analysis ................................................................. 4
   2.1 Current Situation ............................................... 4
      2.1.1 Mission .................................................. 4
      2.1.2 Hardware / software .................................... 5
      2.1.3 Communication ........................................... 5
   2.2 Operational Demands ............................................ 6
   2.3 Human Factors Knowledge ...................................... 6
      2.3.1 Information overload ..................................... 6
      2.3.2 Optimal Workload ........................................ 8
      2.3.3 User Interface ............................................. 9
   2.4 Envisioned Technology ......................................... 12
      2.4.1 Message Relevance Determination ....................... 12
      2.4.2 Content Detection ....................................... 13
      2.4.3 Conclusion ............................................... 13

3 Specification .......................................................... 15
   3.1 Core Functions .................................................. 15
   3.2 Use Cases ........................................................ 15
   3.3 Requirements .................................................... 16

4 Design ................................................................. 17
   4.1 Architectural Design ............................................ 17
   4.2 Technical Design ................................................ 17
   4.3 User Interface Design .......................................... 18

5 Implementation ....................................................... 19
   5.1 Relevance Algorithm .......................................... 19
      5.1.1 LSI ....................................................... 19
      5.1.2 Fuzzy System ............................................ 22
      5.1.3 Bug in the used LSI library ............................. 25
   5.2 Database Library ................................................. 25
      5.2.1 TrexCore ................................................ 25
      5.2.2 Packages ................................................ 25
      5.2.3 Ontology ................................................ 26
      5.2.4 Change of ontology ...................................... 26
   5.3 UserApp .......................................................... 26
   5.4 Ticker ............................................................... 26
   5.5 General Libraries ................................................ 27
      5.5.1 Apache log4j ............................................. 27
      5.5.2 Apache Commons Configuration ......................... 27
      5.5.3 Implementation of the Observer pattern ................ 28
   5.6 Software Improvement Group .................................. 28
J.8  Starting the relevance determination algorithm ........................................... 62
     J.8.1  Changing the fuzzy rules ................................................................. 62
J.9   Emulating a scenario (optional) ................................................................. 62
     J.9.1  Emulating a scenario using Trex ...................................................... 62
     J.9.2  Emulating a scenario using the algorithm program ......................... 63
Chapter 1

Introduction

Project Motivation

During modern military operations the environment is dynamic and unpredictable. There are many actors in the area of the operation, military as well as non-military, friendly and non-friendly. Combined with personnel reductions this creates the need for a flexible and multifunctional military organisation. In an organisation where less people need to do more it is very important to utilise the skills of every individual. Utilising those skills will create a “power to the edge” structured organisation where decision making is also done at lower levels.

Mission efficiency could increase if individuals or groups can anticipate and act more autonomously under changing circumstances and events. To facilitate this, providing relevant information for an optimal Situational Awareness is essential.

Objective

A possible way to improve situational awareness is to send information not just to the specified receiver but also to another group if the information is deemed relevant for that group. The goal of this report is to see how the process of transmitting information can be automated and how this can facilitate anticipation and increase efficiency.

The following main research question has been formulated:

• Can we develop an automated system that scores messages on relevance to a certain person or group, to assist in facilitating anticipation and increasing efficiency?

The following sub questions have been formulated:

• How can message content be used to score the relevance of a message?

• How can message metadata, like distance between receiver and possible receiving group, be used to score the relevance of a message?

• How can a user interface be designed that uses the relevance score in the presentation to the operator?

This project is part of the “Smart Operations” (SmartOps) research programme, which supports the military with systems which have a better “understanding” of the situation.
Situated Cognitive Engineering Methodology

During the execution of this project the situated Cognitive Engineering [2] (sCE) methodology was used. For a large part this methodology has determined how the project was executed in different phases. This final report describes the project following these phases as determined by the sCE methodology.

The sCE methodology has a focus on coming to well justified requirements. The sCE methodology consists of the following phases: derive, specify, test and refine. Figure 1.1 represents the different sCE phases. During the derive phase research on required operation demands, human factors knowledge and envisioned technology is performed. In the specify phase requirements are created, based on the results of the derive phase. Each requirement comes with one or multiple use cases and claims. After building a prototype these claims are verified during the next phase, which is the testing phase. Based on these tests, if needed, the requirements are adjusted. This process of specify, test and refine is an iterative process.

Figure 1.1: The situated Cognitive Engineering process [2].
Reading guide

To support the implementation phase two additional documents have been created to complement the sCE methodology. The extra documents contain the implementation decisions for the built prototype. These implementation decisions can be found in the architectural design document (ADD) and the technical design document (TDD).

Combining the sCE methodology and the extra implementation phases, leads to the following chapters in this final report to describe the project:

Chapter 2 Analysis
Chapter 3 Specification
Chapter 4 Design
Chapter 5 Implementation
Chapter 6 Pilot
Chapter 7 Conclusion
Chapter 8 Recommendations
Chapter 9 Reflection

In the chapter Analysis, the phase derive is documented. The operational demands, human factors knowledge and envisioned technology are discussed. In the chapter Specification, the phase specify is documented. This includes the use cases, claims, requirements and use cases. In the chapter Design, the fundament for the implementation is laid, using the user interface design, architectural design and technical design. In the chapter Implementation, the implementation phase itself is documented. Next, the pilot is described in its own chapter, with the pilot’s design and observations. The final report finishes with a conclusion, recommendations and a reflection.
Chapter 2

Analysis

This chapter describes the analysis done for this project. Following the sCE methodology, this chapter is separated into three main sections. First operational demands is discussed; how is the command structure and how is the current situation now? Second human factors knowledge is discussed; what are the effects of information overload to users and how can a strong user interface be constructed. Finally it surveyed how message filtering and grouping can be implemented by means of technological algorithms. The results of this analysis will be used in Chapter 3 to derive use cases, claims and to set up requirements.

2.1 Current Situation

2.1.1 Mission

This section contains confidential information and was partially left out.

Command structure

The structure of command was deduced after the mission was completed and gives an accurate insight on this. A graph of this command structure can be found in Figure 2.1. In Table 2.1 all abbreviations and their full name can be found.

Figure 2.1: Deduced chart of the command structure during the simulated mission. (Confidential)

| Table 2.1: Abbreviations and definitions of military roles. (Confidential) |

Phases / tasks

Normally the mission structure would be created before the start of a mission. With this simulated mission the mission structure was deduced afterwards and gives a very detailed view of the flow of the mission Figure 2.2. The defined tasks give an indication what the relations are between tasks and the actors that are part of the scenario. How tasks interact and how close they are related can also be observed from the diagram.

Figure 2.2: Mission tasks tree. (Confidential)

Placement

In Figure 2.3 the placement of the different people participating in the simulated mission is shown. Confidential text was left out.
2.1.2 Hardware / software

Confidential section.

Figure 2.4: Confidential screenshot.

2.1.3 Communication

Confidential section.

Figure 2.5: Confidential screenshot.

Message log

All messages are logged in a text file for further investigation. The format can be seen in Figure 2.6. The important factors are the time stamp, user, action, communication group, transmitter and finally the message.

Figure 2.6: Confidential screenshot.

Message build-up

Using the information found in the log we can conduct a diagram (Figure 2.7) which derives all available metadata for a message. The metadata can be used as parameters for a algorithm in order to derive the messages relevance (see also Section 2.4.1).

![Diagram of message metadata](image)

Figure 2.7: A diagram representing metadata for a message.

Other forms of communication

During the operations not all communication is done by text but also speech is involved. Not all communication can be reduced to only use text, however it possible to convert any speech to text by an application, making it possible to use speech as text.
2.2 Operational Demands

In this section we describe the operational demands. The operational demands in our case are the demands that are present during a military mission, that are related to the communication between teams. Below is a list of the operational demands we concluded from our research.

Work has to be carried out in ad-hoc formed teams. In current situations, with cuts in the budgets, teams that have to cooperate during a mission are often formed “ad-hoc”. This means there is no clear command structure at the start of the mission. Tasks and responsibilities are often not determined in advance. While the mission is going on, people will take their responsibilities themselves, and an organisation structure appears.

Self-regulation has to be enabled. In the modern approach during military missions teams do not have a superior officer that decides what the team does. Teams make their own decisions on how to accomplish their objective. This also means every team needs more situational awareness to correctly make decisions.

Different teams depend on each other. Despite teams make their own decisions, they still have to work together to complete the objective(s). This means information has to be passed between the teams.

A lot of information has to be processed. With the current technology and multiple teams working together a lot of information is produced. All this information has to be processed, while the teams are doing their tasks. This all has to be done with limited amount of time available.

Work has to be carried out in a highly dynamic environment. Military operations are complex. During the operation the environment is unpredictable and dynamic. There are many actors during an operation. Therefore military units need to be flexible and able to respond to the changing environment.

[1]

Anticipation has to be enabled. In order to effectively complete a military mission, anticipation on upcoming events could prove very useful. If personal is able to anticipate time can be used more effectively. In moments someone has nothing todo they could prepare tasks they expect to have to do in the future, and allow them to be faster when the task has to be carried out.

Situational awareness. In order to enable this anticipation, personnel has to know what is happening in the environment. Personnel needs to know more than just information about the task(s) they are working with.

Personnel should have a correct workload. It is relatively easy to give every person all information that is available. This however would not work, as personnel does not have enough time to process all the information. Therefore the information has to be filtered in order for the personnel to be able to process it, and thus build up situational awareness.

The user has to trust the information supply. The user’s dependence on the system will be high; a large part of communication will be processed by the system. This dependence can only be accepted by the user if they trust the information presented to them and trust the system they work with. This means that information should not be inaccurate due to the system and that the system should be stable. For the prototype these requirements are not hard, however, during design these requirements should be kept in mind for a future system.

2.3 Human Factors Knowledge

2.3.1 Information overload

Information overload is a situation in which a person can no longer accurately execute their task because they must process too much information. Information overload can present itself in many contexts, where information processing is part of the execution of a task. Information overload has many unwanted effects on the effectiveness of the execution of the task. According to van Lieburg et al. [3], the following are symptoms of information overload:

1. Reduction of integration capacity. The user no longer has the general overview.
2. Reduction of differentiation capacity. This leads to fuzziness in function, task and role.
3. Limitation on information processing. A user can compensate for information overload by selectively processing information sources. This can lead to a loss in comprehension of the current situation.

4. Too much filtering. Instead of selectively processing information sources, the user will selectively process information on a case-by-case basis. In an information overload situation, the user can filter out too much information.

5. Irritation. Due to a lack of control, the user can become irritated. This can influence other people and lead to a downward spiral.

6. Physical symptoms. Due to stress caused by information overload, the user can have a higher heart beat, have an increase in sweat production and will have higher brain activity.


8. Paralyses. The user will postpone decision making.

9. Information discrepancy. Information processing is out-of-sync with information generation.

10. Fixation. A user will lose long-term planning and instead focus too much on short-term planning.

11. Cognitive lock-up. The user can be locked-up in one task for too long.

12. Hubris. When a lot information reaches a user, the user becomes more confident making decisions, even though not all information is taken into account.

13. Ad-hoc decision making. To make fast decisions, heuristics are used by users.

14. Hyper-vigilance. Due to an overload of information, the user can no longer see the difference between values and properties of available information.

15. Confusion and stress. When a user feels overloaded, this can lead to stress and confusion.

Several factors determine the level of information overload. Van Leiburg et al. [3] determined the following factors in a Command and Control environment:

1. Uncertainty. When not enough useful information is available, uncertainty arises.

2. Inconsistency. Inconsistency arises when available information can be interpreted in several ways, leading to uncertainty.

3. Familiarity. How good new information agrees with the current context determines its familiarity. Unfamiliar information can be judged to be irrelevant by the user.

4. Intensity. Intensity is determined by both the speed information arrives at the user and the relevance of the information.

5. Quantity. Quantity is the quantity of information units still to be processed.

6. Presentation medium and form. Presentation media and forms determine how efficient an user can process information.

7. Timeliness. Since many task are time critical, this can have an effect on information overload.

8. Task-saturation. When a user has to execute too many tasks, they will no longer be able to finish all tasks.

A model to assess information overload based on the most important factors was developed by Neerinck et al. [4]. The model can be described using Figure 2.8. As can be seen in this figure, cognitive task load depends on task switches, percentage time occupied and the level of information processing (LIP). Furthermore, some symptoms of information overload are indicated in the figure: vigilance, cognitive lock-up and complete overload. Underload and optimal workload are also indicated.
2.3.2 Optimal Workload

According to van Lieburg et al. [3], in a Command and Control (C2) environment, four different functions can be differentiated. These are Situational Awareness (SA) \(^1\), Threat Assessment (TA), Decision Making (DM) and Direction and Control (DC). From input, Situational Awareness is created. From this (mental) image, a Threat Assessment can be made. Based on this TA, decisions are made. Based on these decisions, Direction and Control is given to actors. This is visualised in Figure 2.9.

When too much information arrives at the bottom of the sensors, the user cannot make an accurate SA model. Accordingly, no accurate TA, DM and DC will occur, since not enough attention can be given to these functions. Furthermore, TA, DM and DC (indirectly) depend on the SA, so if the SA is not accurate because information cannot be processed effectively, neither will TA, DM and DC be accurate.

It can be concluded that the sensors of the user should be stimulated so optimal workload will occur. This will lead to optimal task execution as described in the C2 model. Optimal workload occurs based on the information load model, described previously.

\(^1\)Different models for Situational Awareness exist. These models are sometimes more relevant for certain areas. The model from Leiburg et al. [3] is suited for a C2 environment. Generally, two SA model types exist. According to Salmon et al. [3], “The main point of contention between theoretical perspectives lies in whether SA refers to the processes employed in achieving and maintaining it or to the end product derived as a result of these processes.” However, these two types of models can be combined. In such a model, the product model is inserted in the process model. The process model will give a feedback loop between the user and the environment. The product model will give different levels of SA.
Optimal workload can be stimulated by only giving relevant information. This can be done automatically using filters. Filtering influences the intensity factor from the information overload factors. In the reduced model from Figure 2.8, time occupied is reduced.

### 2.3.3 User Interface

#### What makes a good UI

User interfaces are found ranging from very poor to the very finest. In this chapter we want to capture what is the cause of these big ranges in quality of user interfaces. To start out, a good user interface requires a number of key factors to be covered as shown in Figure 2.10 [6].

For a C2 environment, it is important that the user can have Situational Awareness in a short time, which is kept up-to-date. To reach this, the data presented should match the user’s mental model of the situation as closely as possible, at a level of understanding as high as possible. As Hutchins et al. [7] state: “Graphic presentations should (1) reduce the amount of mental computation required to perform tasks; and (2) allow users to spend less time searching for needed information.”

<table>
<thead>
<tr>
<th>Clear</th>
<th>A clear interface helps prevent user errors, makes important information obvious, and contributes to ease of learning and use.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistent</td>
<td>A consistent interface allows users to apply previously learned knowledge to new tasks. Effective applications are both consistent within themselves and consistent with one another.</td>
</tr>
<tr>
<td>Simple</td>
<td>The best interface designs are simple. Simple designs are easy to learn and to use and give the interface a consistent look. A good design requires a good balance between maximizing functionality and maintaining simplicity through progressive disclosure of information.</td>
</tr>
<tr>
<td>User-Controlled</td>
<td>The user, not the computer, initiates and controls all actions.</td>
</tr>
<tr>
<td>Direct</td>
<td>Users must see the visible cause-and-effect relationship between the actions they take and the objects on the screen. This allows users to feel that they are in charge of the computer’s activities.</td>
</tr>
<tr>
<td>Forgiving</td>
<td>Users make mistakes. User actions should be reversible. A good interface facilitates exploration and trial and error learning.</td>
</tr>
<tr>
<td>Provide feedback</td>
<td>Keep the user informed and provide immediate feedback. Also, ensure that feedback is appropriate to the task.</td>
</tr>
<tr>
<td>Aesthetic</td>
<td>Every visual element that appears on the screen potentially competes for the user’s attention. Provide an environment that is pleasant to work in and contributes to the user’s understanding of the information presented.</td>
</tr>
</tbody>
</table>

Figure 2.10: UI Needs [6].

Important to note is that although a user controlled is preferred due to the complexity of the environment it is important that the user’s attention is guided by the system. This kind of user interface is called an “Attentive user interface”. By using visual affects to guide the user or if an event is important a interrupt is caused by the system.

#### Design

The construction of the current user interface is the basic T where the user continuously scans the four information screens. To design a new user interface the focus will be on the following topics. First up is the colour schematic to ensure the user is focused on the right thing at the right time. Secondly we
will discuss the different approaches in creating interaction between the user and the screen. Augmented reality and its benefits and drawbacks is the last topic.

**Colour** When designing a graphical user interface, colour is a very important factor to take into account. Each colour has some stereotypical meaning and combining those can create a typical feel to an interface. See also Figure 2.11.

![Colour Meanings Table](image)

Figure 2.11: The meaning of different colors.[8]

Important to note is that these colour meanings pertain to the western culture. A quick example: In other cultures the colour red has quite different meanings. In China red means happiness but in Egypt red means death.

A second important feature of colour is that the combination of background and foreground colours can help the user but just as easily wear the user out. Figure 2.12 gives an indication of what colours can be used in conjunction with each other.
Controls

A tool to improve the experience of a user with the user interface is changing the way he has to interact with the system. There are many different approaches to choose from each with different advantages and disadvantages.

Mouse and or keyboard  
The basic controllers of the modern computer at home or at work have the advantage that almost everybody has experience with these devices. Although that most users are experienced these devices are not very efficient. Working with both mouse and keyboard often has the user switching his mouse hand back and forth. Besides the switches the standard “QWERTY” keyboard is very inefficient to type on.

Touch  
Using your fingers or a stylus to make clear to the interface what you want is a very intuitive way of working. The iPad and iPhone are both examples of how easily accessible a computer can be even for non-experienced users. Having the option to enlarge a window by actually pulling the edges is easier than trying to sweep with the mouse.

Speech  
Talking to the computer sounds like science fiction but is more and more incorporated in daily life. The car company Ford is already pushing its new speech recognition to allow the driver to control his navigation, phone and entertainment system. Speech recognition is a very powerful tool if the user trusts the computer. A few false positives, where the device does something else than was intended, will make the user long back to his mouse and keyboard.

Eye detection and mind reading  
Brainwave detection and eye detection are very new methods for controlling the computer. The systems are still being researched and improved upon and are mostly used in a gaming environment where there is little consequence from failing. Important to note is that
the ability of a user to operate this kind of controller scales with his ability to imagine. A user needs to visualize something to make the computer do that. This means visualizing and actually believing to move a cup with your mind is needed to make a cup move on the computer. In practice it showed that children are better at operating than adults. Adults have defined a lot of subconscious barriers about what they can and can’t do, resulting in lesser results.

2.4 Envisioned Technology

As described in Chapter 1, the main project objective is to determine the relevance of a message for a specific person. In this section several possibilities for scoring and grouping messages are explored. Combining one or more of these methods may be the core of the algorithm which determines the relevance for a message. Determining a relevance score is done over variables and constraints. These variables are based on the message metadata, which can be found in Figure 2.6. Messages grouping is done based on their content. For both determining the relevance and content grouping we will look into and autonomous- and non-autonomous learning methods.

2.4.1 Message Relevance Determination

A Bayesian network can be used to find probabilistic relationships among variables of interest [13]. A Bayesian network is usually represented by an acyclic directed graph. Each node represents a variable. Each node has a conditional probability table representing the effects of the parent nodes on this node. Calculation the inference of variables is done by standard Bayesian reasoning, however calculating this for the graph a NP-Complete problem[14].

An expert system [15] can be used for filtering messages. A knowledge base, i.e. a set of rules, is created by one or more experts. When a message has to be evaluated the system executes the required set of rules, using an inference engine, in order to derive a result. Expert systems can be applied in small domains, and they can work with incomplete data. Generally, it is very easy to make changes in the rule set. However the system can make mistakes; this happens when the knowledge base contains errors, or when data is interpreted incorrectly.

![Basic structure of an expert system.](image)

Figure 2.13: Basic structure of an expert system.

A refinement on expert systems is to use fuzzy logic [16]. A classical expert system evaluates with Boolean expressions only (true and false values); however a fuzzy expert system uses fuzzy expressions. In an expression all literals have a value in a certain range, e.g. from 0 to 1. Also, a value can be in different classes. For example \( x \) is 0.1 small, and 0.6 medium. A fuzzy expression is applied on the fuzzy variables in order to get a new fuzzy result. The fuzzy inference engine can be implemented in several ways. Usually the inference mechanism is implemented with the Mamdani- or the Sugeno-algorithm.

A decision tree, or decision tree learning, can be used to classify information. The tree is built from the root node [17]. Each node tests an attribute, each branch represents the attribute value. The leaf nodes represents a classification. In order to classify an element \( x \), its path in the tree followed starting from the root node, ending in a leaf node. A decision tree can be trained. A decision tree has some
limitations [18]. For example decision trees can not handle noisy or incomplete information. Continuous data, such as a temperature, has to be grouped into ranges, which may hide important patterns. Also it can not evaluate more than one variable at a time, i.e. each edge tests one attribute.

A technology based on biology is Genetic Programming. Genetic programming is a technique that evolves a program as a solution for a certain problem. A program consists of a predefined set of functions, which are the building stones for the program to be evolved. The evolution of a Genetic Program is guided by its fitness function. Genetic Programming does not require the user to specify in detail what the solution should look like, it evolves the solution based from a high level description [19]. Therefore Genetic Programming can effectively be used in a situation where the user does not exactly know how a solution has to work, but can define what the solution must be able to do. A successful example is the design of electrical circuits [20]. To build the program that does the evolution you need to do a lot of research, as you need to specify the function set.

Also one may consider case-based reasoning. Case based reasoning uses previous cases in its reasoning [21]. Since we have no available past solutions to be used we will not further consider case-based reasoning.

2.4.2 Content Detection

Another part of the problem our algorithm has to solve is content detection. By content detection we mean determination of extra metadata inside the content of the text messages. Our main focus lies on the detection of the topic of text messages.

A method to determine the content of a text message is Latent Semantic Indexing (LSI). LSI is an indexing method based on the mathematical technique called Singular Value Decomposition (SVD) [22]. It determines the relation of terms (words) and a collection of text. At the moment LSI is used for multiple purposes, like spam filters and automated document classification. In our case we could use LSI to index a library of words based on the content of chat messages, and then match the words from new text messages to the words in the library.

The main advantages of LSI are that it has not much problems with single words that have multiple meanings. Also it is very tolerant to words spelled on a different way. It does not need the words to be in a sentence [23]. Many chat messages do not contain a sentence, especially when written in a hurry.

The largest disadvantage of LSI is the computational power required to build the index. Since we do not have much data at all that poses no problem for our experiment. Another disadvantage is that it is sometimes hard to find an optimal dimension for the SVD Algorithm. More dimensions produce more specific classifications while fewer dimensions give a more general one [24].

Neural networks is a technology based on the biological neural network system. The technique uses artificial nodes to simulate biological neurons. Neural networks are used to transform input data into a data more easily handled by an algorithm. Neural networks are mainly useful when the input data is too complex to design the function in the ordinary way. The most well known example is face recognition, but it can also be used for data mining and spam filters [25], which approaches the problem in our project. One of the main issues with neural networks is that it requires a lot of training. For the training diversity is important. Because we have to work with a dataset of approximately 300 items based on 100 different text messages, we do not expect it is possible to train a well-working neural network to use in our project.

2.4.3 Conclusion

After we researched the methods described above we draw some conclusions. In Table 2.2 all discussed methods are listed. The main properties we believe are relevant for this project are listed as columns. The table represents a map of the properties the system can handle. Can the system handle fuzzy data, are the results from it interpretable (i.e. do we know what happens when the method is being used), is the system is trainable, how big should the learning be, and can the system easily be adjusted by others.
We believe that a fuzzy expert system is best suited for message filtering. We are dealing with a small set of rules to determine relevance of a message. These rules can be easily put into, and maintained by an expert, without the need to retrain the program. Since we are dealing with fuzzy metadata, a fuzzy expert system is more suited than a default expert system. A Bayesian network would cost more time to set up because of the already present experience in the group with fuzzy expert systems. Also genetic programming would not be a good solution for message filtering because it requires a large data-set for training, which we do not have. Also as described earlier, for a genetic algorithm one needs to specify the functions (building stones) for the program. Defining the building blocks requires a lot of in-depth knowledge of the problem itself, which we do not have.

Content detection, as one parameter for the filtering algorithm, then can best be determined by using LSI. Technically we have no proof that LSI works better than using a neural network. Tough for LSI there is a framework available ready to use, and it is currently being used successfully for spam filtering and search algorithms for documents. For neural networks it is harder to find examples of applications within the topic detection scope.

<table>
<thead>
<tr>
<th>Method</th>
<th>Fuzzy data</th>
<th>Human interpretable result</th>
<th>Trainable</th>
<th>Learning set</th>
<th>Adaptability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayesian network</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
<td>Low</td>
</tr>
<tr>
<td>Expert system</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>-</td>
<td>Low</td>
</tr>
<tr>
<td>Fuzzy expert system</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>-</td>
<td>Medium</td>
</tr>
<tr>
<td>Decision tree learning</td>
<td>No</td>
<td>Yes</td>
<td>?</td>
<td>?</td>
<td>Medium</td>
</tr>
<tr>
<td>Genetic Programming</td>
<td>Yes</td>
<td>Depends</td>
<td>Yes</td>
<td>Large</td>
<td>High</td>
</tr>
<tr>
<td>Latent Semantic Indexing</td>
<td>Yes</td>
<td>Depends</td>
<td>Yes</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Large</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 2.2: Overview of discussed methods and their most important properties.
Chapter 3

Specification

In the previous current situation has been analyzed. In this chapter use cases, requirements, claims and core functions are derived and represented. First, the core claims are listed. The core functions will serve as a starting point to specify the requirements. The use cases will provide the context for the requirements. Each requirement has one or more claims, which can be used to measure if the requirement has been met.

The requirements, use cases and claims were entered in TNO’s situated Cognitive Engineering Tool, which was developed by Wytse Posthumus. This is an online application, which was used to export the requirements, use cases and claims to Latex, so they could be entered in this final report.

3.1 Core Functions

In this section the core functions of the relevance system are described, which are a result of our analysis. Core functions are high-level requirements, from which functional requirements are derived in the next section. The core functions are:

1. Automatically determine the relevance of text messages during a military mission; This core function shall determine the relevance of information for a specific person during a military mission. Every text message is processed by the relevance system, and relevance will be determined for the HDO.

2. Filter information bases on the relevance; This function enables the filtering of information, based on the relevance determined in the previous function. The operator is able to modify the relevance filter during a mission, to increase or decrease the amount of messages shown.

3. Clearly show the text messages to the user; The relevance system also focuses on showing the text messages in a clear way. For example, ordering can influence this clarity. Ordering can be based on different properties of the messages, such as time, subject and relevance.

3.2 Use Cases

The purpose of use cases is to contextualize the requirements. According to Necrincx et al. [2], “A large set of use-cases, contextualizes the requirements, indicating in what kinds of situations any given requirement applies”.

Compared to scenarios, use cases are formalized scenarios. “Use cases describe the general behavior requirements for software systems or business processes [...]”.

Uses cases can be found in Appendix A.

The use cases are defined in a specific format. First, the actors of the use case are defined. These are the users whom play a role in the use case. Next, the goals of the use case are stated. This defines what goal is achieved when the use case is carried out. A short description of the use case follows. Next, the pre-conditions are stated. The pre-conditions indicate the requirements that have to be fulfilled before the use case can be carried out. The post-conditions are the guaranteed conditions after the use
case has been carried out. After the post-conditions the related requirements follow, to indicate which
requirements are contextualized by the use case. Finally, one or multiple sequences is specified. These
sequences give the interaction between the actors and the system when the use case is carried out.

3.3 Requirements

Requirements are the functions that are to be implemented in our relevance system. For each requirement
we have measurable claims to justify the requirement. Claims should therefore be concrete and testable[2].
As can be seen in Figure 3.1 claims justify requirements while use cases give context to requirements.
Claims indicate what the positive and negative aspects are of a requirement. Use cases show in what
situations a requirement applies.

Requirements can be found in Appendix B. The requirements are specified in a specific format. First,
a short name of the requirement is specified. Then, a short description of the requirement follows. Next
are the claims, which justify the stated requirement. For each claim, the advantages and disadvantages
are noted. Finally, related use cases are mentioned, which give the context for the requirement.

Writing down the claims which justify the requirements has an advantage. If circumstances change,
and advantages and disadvantages of claims change accordingly, this can have implications for claims
itself and therefore for requirements which are justified by claims. Requirements can then be adjusted
accordingly to changed claims.

Figure 3.1: Claims, use cases and requirements [2].
Chapter 4

Design

Based on the core functions, use cases and requirements discussed in Chapter 3, different designs were made. First, the Architectural Design was made. This is a high level design, defining the system components and their interaction. The design of these individual system components is refined and specified in the Technical Design. Finally, the user interface design of the UserApp, one of the relevance system components, is described in Section 4.3. Both the architectural design and the technical design can be found in their own documents, but are briefly discussed in the upcoming sections.

4.1 Architectural Design

The architectural design for the prototype is described in the Architectural Design Document (ADD). The ADD is a separate document. It intensively discusses the architectural decisions made for building the prototype. These decisions define the system components, the communication between the components, how to store data, and by which libraries the components of the algorithm are implemented. The information gathered in the research phase is used to fill in the system components and link the components. By creating the ADD we ensure that all components that are needed are developed. In this section the results of the document are listed. Please refer to the ADD for more details and reasoning.

The prototype will consist of different stand-alone software applications. First there is a relevance algorithm, which uses the techniques of LSI and fuzzy expert systems in order to rate a message. There is a UserApp presenting the messages based on their rate. A message simulator reads a log file and puts these messages in the system. In order to simulate real life values a ticker, profile manager and user data emulator can be implemented. In order to be compatible with TNO prototypes all these components communicate directly to a database. This enables others to easily re-use and integrate the work done in this project for other projects. The database used is Sesame. Sesame is an RDF based database, meaning that there are no specified columns. Each relevance system component using this database can define needed fields, and others may use those fields or may simply ignore them. In order to communicate to this type of database, a new library is developed. This library will be compatible with other software applications using TrexCore, which is used for some TNO prototypes.

4.2 Technical Design

The technical design document (TDD) contains all details regarding the design of the different software components. The TDD refines the design of the system components from the ADD. For the database component, the TDD contains the database ontology. For all other system components which will be implemented and the database library, class diagrams are made. Continuing on the class diagrams, the used design patterns are stated. The last paragraph of the TDD contains the specifications of all classes.
4.3 User Interface Design

Several Graphical User Interface designs are made. The designs differ in how information is displayed and used to lay-out user interface components. For example, each message has a relevance. This relevance can be displayed to the user by giving the message a color based on this relevance. Another option is to position messages differently based on relevance. Whether a message is addressed to the user, is also information. This can also be used to give different colors based on whether the message is addressed to the user.

All these factors allow many different graphical user interface designs. The first step was to come up with a small set of graphical user interface designs, based on our own opinion on what might work. The different designs are described in Appendix C.

Military experts were consulted to give their opinion on these designs. On 10 May 2011 three experts were interviewed: one officer and two non-commissioned officers of the Royal Netherlands Navy. The results of these interviews can be found in Appendix D.

From these results it can concluded that preferences for the user interface are very personal. The idea behind the user interface is the same: the information should be shown in a clear way, and it is useful to be able to see to what topic a message belongs. About how this should be done all operators had their own preferences.

Since the first priority is on the relevance determination, it was decided to first build a very simple graphical user interface. The first proposed design from Appendix C was chosen. It consists of two columns. The left column displays messages addressed to the user and the right column shows relevant messages, ordered by relevance. This basic design allows a clear demonstration of the relevance determination.
Chapter 5
Implementation

This chapter describes the implementation of the different software applications as defined in Chapter 4. Difficulties encountered during the implementation are also described, including the solution to these difficulties. Moreover, the tools used for implementation are listed, such as the different software libraries used.

The relevance system consists of several separate stand-alone software applications. Because Sesame is used, as described in the ADD, the database is a central component between these software applications. All applications react on changes in the database and then perform their tasks. See Figure 5.1 for an overview of the applications.

When a new message enters the database it has no relevance value. This means for now the UserApp does nothing with the message. The algorithm detects the message with no relevance and starts calculating the relevance value. See Figure 5.1. The LSI and Fuzzy Rules are explained later. After the Relevance Algorithm determined the relevance value it is stored in the database.

Now the UserApp detects the message with the relevance value and the message is shown to the user. The user can give feedback over the relevance of the message. This feedback is also stored in the database, and is used for the relevance determination of new messages.

In the next sections the different components are described in more detail. In Appendix E a test plan is available which describes how the implementation is tested.

5.1 Relevance Algorithm

The relevance algorithm is the part of the relevance system that determines the relevance of messages for an operator. When a new message is passed to the system, it is first added to the LSI System. The LSI System produces three values. It determines the average relevance of messages that are semantically similar, the percentage of the similar messages that are directly addressed to the operator and a value for the reliability of these two results.

The results from the LSI System then are passed to the fuzzy system. The fuzzy system applies the given rules for the operator on the input variables, and produces a value for the relevance. The fuzzy system produces two relevance values, one that does take the age of messages in account and one that does not. The values are stored in the database so the UserApp can use them. See Figure 5.1 for an overview of the algorithm in the system. In the following two sections the LSI and fuzzy system are described in more detail.

5.1.1 LSI

The LSI index produces a vector representation for each message. With these vectors the distance between each message can be determined. In this way, it can be determined which messages are semantically similar and which are not.

Different methods to build this LSI index were tested to see what method was best. To explain this best method, the first assumption is that a list of messages with a relevance value to start the system up. The startup of the system will be described later.
When a new message arrives it is added to the LSI index. By doing so, a distance is obtained between the new message and every other message in the list. Based on the experiments a filter-value was chosen, which select every message with a relevance larger than the filter value. This results in a list of similar messages.

Something must be obtained which can be used in the fuzzy system. It was decided that two properties
should be used as input for the fuzzy system. First, the average relevance of all similar messages is calculated. This is used as the first parameter for the fuzzy system. Second, the number of similar messages that had the operator for the relevance for as direct receiver is counted. This number is divided by the total amount of similar messages, so a percentage for how much of the similar messages were directly addressed to the operator is obtained.

Now two values can be used in the fuzzy system. Sometimes the LSI index did not provide much similar messages. In these cases, the average relevance and the percentage of direct messages could be based on a single message. If that single message was directly addressed and had a high relevance the result would be that the new message also had a high relevance and a high percentage (100%) of directly addressed messages. This might be true, but the result is not very reliable as it is based on a single result.

Therefore, a third value was added: the reliability. The reliability is the number of messages on which the result is based. This is the number of similar messages from the LSI index. The value for reliability now also can be handled in the fuzzy system. See figure 5.2 for an overview.

Figure 5.2: Overview of the LSI system.

To come back to the startup of the system: as was stated, the reliability value for the system is calculated, which is based on the amount of messages that are similar. When the system starts up,
there are few messages in the system. This means the amount of similar messages is always lower, and so is the reliability. The fuzzy system now will not weight the LSI result very heavily in the relevance determination, and the relevance has to be obtained by other means. When more messages are added and the system is getting trained better, the amount of similar messages also increases, and so the reliability parameter increases. Now the fuzzy system will add more weight to the LSI results.

5.1.2 Fuzzy System

Ruleset Definition

After the LSI system calculated the relevance and direct addressed percentage for the similar messages the fuzzy system is executed. In addition to the input variables from the LSI system, the fuzzy system also uses other variables, like the distance between the operator and the sender of a message.

For the fuzzy system a library called JFuzzyLogic is used. JFuzzyLogic is an open source Java library, distributed under the GPL and Apache licenses. JFuzzyLogic can be downloaded for free from [http://jfuzzylogic.sourceforge.net](http://jfuzzylogic.sourceforge.net). JFuzzyLogic can interpret a fuzzy ruleset defined in Fuzzy Control Language (FCL).

FCL is used to to describe input variables, output variables and rulesets for the output variables. The first thing in a FCL ruleset is the definition of the variables. There are two types of variables, the input and the output. For example:

```plaintext
VAR_INPUT
  reliability : REAL;
  relevance : REAL;
END_VAR

// Define output variable
VAR_OUTPUT
  result : REAL;
END_VAR
```

These rules define two input variables: reliability and relevance and an output variable: result. Next are the fuzzy terms for the input variables. Terms are used to label ranges of input variables. For example, the terms relevant and irrelevant can be used for the input variable relevance. For each input value, it is defined how much this term is true. See Figure 5.3 for a graphical representation. This way, the variables are fuzzified.

```plaintext
// Fuzzify input variable 'Reliability'
FUZZIFY reliability
  TERM reliable := (0, 0) (1, 1) ;
ENDFUZZIFY

// Fuzzify input variable 'Relevance'
FUZZIFY relevance
  TERM relevant := (30, 0) (60, 1) (100,1) ;
  TERM irrelevant := (0,1) (30,1) (60,0);
ENDFUZZIFY
```

The inverse has to be done for the output variable: the variables need to be defuzzified. Given how much certain terms are true, an output variable is calculated. Several methods are possible, the one used by us is the Centre of Gravity method (CoG). This method can be understood best with piles of sand. For each fuzzy term, the position of the pile of sand on the output variable is defined. The height of this pile of sand is determined by how much this term is true. The CoG is then calculated, to be used as output variable. See Figure 5.4 for a graphical representation. The FCL definition looks as follows:

```plaintext
// Defuzzify output variable 'tip'
DEFUZZIFY result
```
TERM irrelevant := (0,0) (5,1) (10,0);
TERM moderate := (10,0) (15,1) (20,0);
TERM relevant := (20,0) (25,1) (30,0);

// Use 'Centre of Gravity' defuzzification method
METHOD : COG;

// Default value is 0 (if no rule activates defuzzifier)
DEFAULT := 0;

END_DEFUZZIFY

After the definition of the variables and terms comes the core of the system, the “ruleset”. In FCL multiple rule blocks can be defined. In the example only one rule block is used. First the methods for the “Aggregation”, “Activation” and “Accumulation” are defined.

Aggregation determines how a condition is calculated. A condition is stated with AND and OR. Since fuzzy variables are used as operands, the boolean definitions of AND and OR are not sufficient. The definition for AND was chosen to be the minimum. This implies that OR is calculated with the maximum of the operands.

Each rule will draw a conclusion, based on a condition. This conclusion is a fuzzy variable. How much this conclusion variable is true due to the condition, is determined with activation. It was decided that product is used for activation. The condition will be multiplied with a constant to determine the conclusion.

Multiple rules may come to the same conclusion. For example, multiple rules may draw the conclusion that the result is relevant. Accumulation determines how these same conclusions are combined. The sum of the conclusions was chosen for accumulation.

After the definition of the methods the rules are defined. In the example, two rules are defined. The definition is very intuitive and the example below should speak for itself.

RULEBLOCK No1

// Use 'min' for 'and' (also implicit use 'max'
// for 'or' to fulfill DeMorgan’s Law)
AND : MIN;

// Use 'min' activation method
ACT : PROD;
// Use 'max' accumulation method
ACCU : SUM;

RULE 1 : IF relevance IS relevant AND reliability IS reliable
THEN result IS relevant;
RULE 2 : IF relevance IS irrelevant AND reliability IS reliable
THEN result IS irrelevant;
END_RULEBLOCK

Now that the fuzzy system is defined, the result for any input can be calculated. Two values for the reliability and the relevance are made. Say the relevance is 40 with a reliability of 0.3. See Figure 5.5.

![Figure 5.5: Example values.](image)

Now the fuzzy rules can be applied. In this case both the rules apply. The message is reliable (since it always is a bit reliable) and the relevance is both relevant and irrelevant. Now for the AND the minimum is used. In this case the minimum for relevant (0.4) and reliable (0.3) is the reliability, thus the minimum is 0.3. The same is the case for irrelevant (0.2) and reliable (0.3).

For activation, the product method is used. So the result value (from Figure 5.4) is multiplied by the minimum value. This means the result value relevant is multiplied by 0.3. The same is done for the result value of irrelevant. All these values are combined in a single graph. The result is shown in Figure 5.6. Note that the maximum of the result has a value of 0.3.

![Figure 5.6: The result graph.](image)

To get the end value, the method defined in the defuzzify block of the return value is used. Here the method was defined as the Centre of Gravity. In this case, as seen in Figure 5.6 the CoG is 15. This is the final result. Would no rules apply, no value of relevance would appear in the graph, and the default value is used.

In our application the ruleset of course is much larger. By applying the ruleset to the values obtained from LSI and the metadata of the message a value for the relevance is obtained. Two rulesets for two different values are defined. One does take the age of the message in account, the other does not. These values are stored in the database, so the user interface can use it.

The fuzzy variables and the description how they are used in our ruleset can be found in Appendix G. The fuzzy ruleset we used to determine relevance for the HDO and does not take message age in account.
can be found in Appendix H. The ruleset for the HDO that does take message age in account can be found in Appendix I.

5.1.3 Bug in the used LSI library

As described in the Architectural Design Document (ADD), the S-Space Package by airhead-research was used as a library for LSI. During the development of the Relevance Algorithm some unexplainable LSI results occurred. It was found that the similarity between two of the same messages was not correct. In some way, the LSI vectors of messages did not correspond with that message. It took some time to find out that this was not a bug in our own software.

The S-Space library was tested with a simple test case. First, it appeared that the library functioned as specified. However, when the messages were changed, altering the number of unique words and the number of messages, suddenly the message vectors were scrambled. The bug was fixed by using an older version of the S-Space package. Because the bug was hard to produce, it took quite some time to find and fix this bug. Such a bug was also unexpected in the S-Space package, because the used LSA implementation was defined as a Tier 1 Algorithm. About Tier 1 Algorithms the airhead-research website states: “These algorithms have been experimentally tested and validated.” [26]

5.2 Database Library

5.2.1 TrexCore

As described in the ADD, TNO Mutual Empowerment’s TrexCore is used. TrexCore is a system for quick prototype building. It enables multiple prototypes to share their data over one single database. In this way a new prototype can easily use another earlier build prototype. It was decided that the relevance system is going to use TrexCore. Unfortunately there was no Java implementation of TrexCore, since TNO has only developed a C# implementation of TrexCore. A new Java library had to be written to be compatible with the existing version of TrexCore. After an interview with the TrexCore maintainer, a guess on the needed amount of time for writing this library was made to be two days extra work. During the implementation it became clear that much more time and effort were needed to write these libraries; however it was decided to continue developing these libraries. The libraries are well designed and successfully implemented, following the design decisions in the TDD.

5.2.2 Packages

As decided in the ADD, the TrexCore Relevance (TCR) package is responsible for connecting the database and mapping data triplets to objects. TCR is set up to be generic for multiple software applications, including applications out of the project scope. Each software application using TCR needs to specify own data models; and these models are stored in an additional package (rdfmodels). It was decided that TCR does not handle relations. Supporting relations will add some complexities, like preventing loops in relations (A has a relation to B, and B has a relation to A). Other relation-complexities come in how far relations could be loaded; for example a message points to a user, which points to n other messages also pointing to some receivers. Finally when using relations in TCR means also implementing a way to keep data persistent between the database and several instances of an rdfobject in the relevance system.

To overcome these relation-bookkeeping problems it was decided to write an extra layer. This layer is called the Overlayer TrexCore (OTC), as seen in the TDD. OTC makes higher level data-objects possible. These higher level objects will contain one or more TCR-data model objects. These OTC models know how to follow relations and methods can be implemented to load a relation-object from the database. Special query objects can be created to select the correct objects from the database.

The TCR and the OTC packages are implemented as described in the ADD (see Sections Data Storage, Sesame libraries, and component communication). The TCR and OTC packages are also implemented as described in the TDD (see Section Class Diagram).
5.2.3 Ontology

The ontology described in the TDD defines of what form the data is stored in the database, and so defines what TCR will receive when executing queries. Representing the ontology has been the main difficulty. In the beginning Google Draw has been used to draw up the ontology but this resulted in some confusing pictures. After some research we discovered a tool named CMap\(^1\) that provided the necessarily functionality to create crisp and clear ontology figures. During the project the ontology has been redesigned a number of times to be compatible with the actual data as can been red in the following subsection.

5.2.4 Change of ontology

During the development of OTC it became apparent that the data set of the scenario provided by TNO could not be mapped to the ontology as defined in the ADD. The location of the senders and receivers was not available in the data set. Instead, in the data set for each message, several distances attributes are provided. These attributes are the time distance with the author and receivers, the social distance with the author and receivers and the task distance with the author and receivers. All these distances are calculated for the HDO.

The position of the sender and receiver could not be determined with this data set. It was therefore decided to add these six distance attributes for the HDO to the ontology of messages. This is not generic for other users of the system other than the HDO, but it was found acceptable for testing purposes, given the dataset. Moreover, in this way the task distance and social distance can be set manually, without the implementation of a mission task manager and role manager. This is in line with decision not to develop these system components, as stated in the ADD.

5.3 UserApp

As stated in Section 4.3 only one user interface is implemented. The user interface is developed in Java, using TCR and OTC for the database information retrieval. The UserApp makes use of the Swing API to draw Graphical User Interface components\(^2\). Swing was chosen because it was familiar to us and it is a common mature Java API for GUI development.

In Figure 5.7 the result of the user interface implementation is shown. The interface consist of a list of messages. The orange bar indicates the system defined relevance for the message. The radio buttons allow the user to indicate how relevant the message is to the user. This message list is implemented with a Swing Table. Adding the options for user feedback and system relevance indication required some extra work. Implementing them was made difficult, because the exact interaction with this added component and Swing is a bit hard to understand.

Development of the controller was relatively straightforward, since the design was thought out during the Technical Design.

5.4 Ticker

The ticker is the App that takes care of keeping track of time and triggering the flag to evaluate the relevance field of a message and update it if needed.

The ticker utilises a timelastupdate field in the database to calculate if a message needs an update to its relevance score. If an update is needed the tick flag is changed in the database to inform the other system components from the relevance system.

The development of the ticker did was relatively straightforward as the use of OTC meant little time had to be invested in the set up of the database communication.

\(^1\)http://cmap.ihmc.us/
5.5 General Libraries

Several of the libraries used are shared by the different system components of the relevance system, such as between the Relevance Algorithm system component and the UserApp. Libraries shared are Apache log4j and Apache Commons Configuration.

5.5.1 Apache log4j

Apache log4j is an open source Java library to facilitate logging in software applications. Log4j can be downloaded for free from [http://logging.apache.org/log4j/](http://logging.apache.org/log4j/) under the Apache Software License. Apache provides similar logging libraries for other languages such as C++ and PHP.

Log4j has proved itself to be a flexible logging library during development of our project. Log4j can be configured using a configuration file to log in many ways. Each time one needs to log, the importance of the logging statement can easily be stated. This allows log4j to log based on filters, which can be configured per Java package. Logging can take place to a text file, the console, remote sockets, UNIX Syslog, etc [28].

5.5.2 Apache Commons Configuration

Apache Commons Configuration is an open source Java library which “[...]provides a generic configuration interface which enables a Java application to read configuration data from a variety of sources” [29]. Commons Configuration can be downloaded for free from [http://commons.apache.org/configuration/](http://commons.apache.org/configuration/) under the Apache Software License.
Our system components can be configured in several ways. Commons Configuration has allowed us to use an XML file for configuration. The values in this configuration file can be easily read using Commons Configuration. Commons Configuration is flexible and support different formats besides XML. Properties list files, system properties and Windows INI files are examples of other supported formats [29].

5.5.3 Implementation of the Observer pattern

Standard Java classes and interfaces were used to implement the Observer pattern. As stated in the TDD, in the Observer pattern, an Observer observers an Observable. In the implementation the standard Java class Observable is extended, which provides the infrastructure for notification of Observers when an Observable changes. For example, the UIController implements Java’s Observer interface so it can observe UIMessages, which extend Observable. This way, an UIMessage can notify its Observers of changes, such as the UIController.

5.6 Software Improvement Group

During the implementation phase the code has been evaluated by the Software Improvement Group. The Software Improvement Group is a company which “gives insight and monitors the quality of software” by evaluating it. It evaluates software and returns a score on software maintainability.

The Software Improvement Group has rated the prototype code during the implementation, and it will rate the software at the end of the project again. The prototype was first rated at roughly halfway through the implementation. The Software Improvement Group has rated our software with almost four stars, meaning that the prototype maintainability is above average.

The improvement hints given were to reduce the module coupling, and to reduce unit length and complexity. When there is high module coupling, there is some code called relatively often; meaning that when a change in that code occurs, it will result in a lot of changes in other code as well. An above average unit length and complexity means that this code is more error-prone than usual, and unit testing may be difficult. Finally the Software Improvement Group indicated that no unit tests were created.

The Software Improvement Group comments were incorporated into the prototype code. Module coupling is reduced, but may not be completely removed because the database communication components and data representation-models are used throughout the code. The longer methods are split up into several sub-methods in order to reduce their length. Unit tests have not been added, as argued in the test plan (Appendix E.2.1). The final pilot code will be re-evaluated by the Software Improvement Group at the end of the project. Unfortunately this second feedback can not be incorporated in this document because the deadline for both the final report and the Software Improvement check are both at the same date.

---

Chapter 6

Pilot

On 20 June 2011, a pilot was held to quantify our claims and to get general feedback on the system. See Table E.1 for the claims in regard to use cases. The pilot was held with a *luitenant-ter-zee der 1e klasse*. The pilot consisted of the evaluation of the Noordzee scenario, described in Section 2.1.1, with our system.

Before the start of the experiment the user indicated that he liked the clear UI but gave the suggestion to improve by adding a column with the function of the receiver and the sender of a message. During the experiment it became apparent that there were some troubles with the scenario and messages as most of the relevant messages would be already communicated vocally. Also sometimes a colleague of the HDO would help by sending a message for the HDO but this results in the system thinking there is a very relevant message the HDO should read.

The general conclusion of the pilot is that almost all claims have received a good score or have a good response on what should be improved. For the exact results please refer to Appendix F.
Chapter 7

Conclusion

This report describes how to create an automated system that scores messages based on how relevant they are to a certain person, to assist in facilitating anticipation and increasing efficiency. By following the sCE methodology a computer system has been developed to enable this.

- Different techniques were researched for analysis of message content. We concluded that Latent Semantic Indexing (LSI) was the most appropriate for this situation. Messages with similar content are linked with each other using LSI. By using LSI the system can calculate a relevance rating based on the relevance rating of similar messages.

- For relevance determination based on the message metadata we use a fuzzy expert system. A fuzzy expert system uses a ruleset to score a message relevance based on different input variables. This ruleset allows for easy adjustment by an expert. We used the information from an interview with experts to create a ruleset. Next to the message metadata it also uses the relevance score from the LSI part as input variable. Another ruleset was created with the same rules, but which also takes the message age into account.

- Several user interface designs were made which present messages based on their relevance ratings. Some designs also use other message metadata, such as age, in the presentation of messages. In an interview with experts, no clear preference for a specific design could be found. However, all experts indicated they would like to be able to see what messages belong to each other, facilitating some thread indication. It is recommended to research the possibility to use LSI in the determination of grouping of messages by similarity.

By taking these solutions together the main research question is answered. We created an automated system that scores the messages on relevance, based on both the content and metadata, and shows the result to the operator. A pilot conducted with an expert indicated that the developed system has the potential to assist in facilitating anticipation and increase efficiency.

There are however important areas where further research and development can be done. As an example the current UserApp can be improved by better visualising the context of a message by showing the age of a message or the author and receivers of a message. Besides the UserApp if a larger and more divers data set would be constructed the LSI and fuzzy expert system could be better fine tuned. Also distinguishing more types of messages like planning messages or sitreps could help calculate a better relevance score. Finally to let the system better simulate scenarios additional system components need to be implemented. The Mission Task Manager, Role Manager and Position Emulator can give the system the ability to simulate closer to reality.
Chapter 8

Recommendations

The sCE methodology is focused on building software in iterations. This report describes the first iteration in the progress of creating the software system. During the pilot an expert was interviewed. In this way the claims were evaluated, and some new insights were formed. In this section future recommendations for research on the software its scope are listed. At least some of these recommendations could be used in a future research iteration, leading to new, updated, and refined claims.

8.1 User interface

During the pilot it became apparent that the current UserApp requires additional functionality. The expert indicated that it is not possible to see the context of a message: who sent the message, when the message was sent, etc. A better understanding of the context would help better with interpreting the relevance values that are presented. In the future the current user interface could be extended in a way that message context is visible.

In Section 4.3 and Appendix C multiple user interface alternatives are considered. Implementing one or more of them is highly recommendable. Refined implementations of these user interfaces can be used to (re)test the claims in a next iteration, testing whether the user interface is fit for actual usage.

Some claims were not tested during the pilot, as can be seen in Table E.1. An example of an untested claim is the addition of thread indication to the user interface. During the interview in Den Helder (Appendix D), the experts indicated this would be a good addition. We recommend the addition of thread indication to the user interface, using LSI to determine what messages belong to each other. Another not implemented claim that could be a good addition is a way to acknowledge messages.

8.2 Test scenarios

The scenario used during the pilot was the scenario described in Section 2.1.1. The dataset of this pilot used to test the algorithm was not fully optimized. In the scenario it sometimes happens that a colleague of the HDO, like the CCO (Table 2.1), sends a message on behalf of the HDO. This means that the author of the message is set to CCO. The algorithm which determines relevance will then have incorrect data and will incorrectly determine message relevance. A dataset in which the author is set correctly would improve the quality of the test results.

Furthermore, the dataset of the scenario was somewhat limited in size. If the dataset contains more messages, the algorithm will be able to make a better index of the message content. With a better index, the message relevance can be calculated more accurately. Apart from the dataset size itself, it would also be better if the dataset is more diverse. In case of the used dataset of the scenario, this would require the dataset to also include messages from many actors outside the ship. This is especially true since some of these messages have the potential to greatly enhance anticipation and situational awareness. Whether this is true can be tested with such an enhanced dataset.
8.3 Different message types

Different types of messages can be distinguished from each other. For example, a planning message may be more relevant for a longer time for a person than a short notification message. This was indicated by the expert during the pilot. If the system can distinguish these different types of messages, the relevance algorithm can be adjusted to give a better score.

A common type of message is the situational report (sitrep) message. These kind of messages are used to update other (external) actors on the current situation. During the pilot the expert mentioned that sitreps from his ship are not considered relevant, but external sitreps (coming in from the other actors) are relevant. However, when a person is off duty and needs to be updated, the internal sitreps are useful to build up the persons situational awareness.

Future research can be done on the message types, and what effect these different types have for the relevance for different persons.

8.4 Unimplemented system components

As described in the Architectural Design Document, not all system components are implemented. Unimplemented system components are the Mission Task Manager, Role Manager and Position Emulator. If these components are implemented, a more comprehensive scenario can be simulated.

8.5 Recommendations for the Mutual Empowerment project

As discussed in the Technical Design Document, the design of the database library follows the Command Pattern. This allows for easy addition of undo functionality, macro functionality and version management. Macro functionality can be used for recording and playback of scenarios at TNO. This could be a helpful feature for testing programs in the Mutual Empowerment project.
Chapter 9

Reflection

This chapter reflects on the execution of our BSc project at TNO.

9.1 Overall projection organization

Our project’s organisation was mainly determined by the situated Cognitive Engineering (sCE) Methodology. sCE has structured our project in the phases of derive, specify and test. The derivation step ensured that the necessary domain information was known to us, before the specify phase was executed. In the specify phase, sCE has helped us mainly in justifying requirements with claims. This allowed us to structurally reflect on the made claims once the test results from the pilot were known.

We decided to refine the development phase of our software by first making an Architectural Design Document and Technical Design Document. These deliverables were familiar to us, since we have made these kinds of documents before at Delft University of Technology.

The Architectural Design Document (ADD) and Technical Design Document (TDD) have helped to design upfront. This enabled us to define the interfaces between the different system components. This proved to be successful: the integration of the separately developed system components was relatively short. The TDD has also helped to design the individual system components upfront, making the designs better. We believe this was one of the reasons that the maintainability of our code was judged to be above average by the Software Improvement Group.

9.2 Time spending

At the start of our project, we made a general planning for the total project. Most of the time, we were on schedule with this planning. Sometimes we were a few days behind schedule, sometimes we were a bit ahead of schedule. We were able to complete this project on schedule to our satisfaction.

To help us keep better control over our progress we set up a wiki page to plan our tasks and write down what actually was accomplished. This made sure we could cope better with unforeseen problems and manage our resources better.

One request of TNO took more time than expected. As discussed in the ADD, TNO asked us to research the possibility to use Sesame for data storage, since this would be beneficial for other projects at TNO. Researching what Sesame was and how it could be used already took many days. The decision was made to make our own database library to link with Sesame, for reasons discussed in the ADD. The implementation of this library also took more time than expected. However, this has resulted in a library with a good design, which TNO can use for future development.

The extra time spent in database technology has required us to spend less time on the user interface. We would have liked to spend more time on the user interface, but we also believe the conceptual user interface designs can be used for future developments.
# Appendix A

## Use Cases

<table>
<thead>
<tr>
<th>UC 1</th>
<th>Reading Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Operator A.</td>
</tr>
<tr>
<td>Goals:</td>
<td>Build up knowledge about the situation (situational awareness), by reading relevant messages.</td>
</tr>
<tr>
<td>Description:</td>
<td>Someone has spare time and decides to read up some information. The person opens up the application to read filtered relevant information.</td>
</tr>
<tr>
<td>Pre-Conditions:</td>
<td>Operator A has spare time between his tasks.</td>
</tr>
<tr>
<td>Post-Conditions:</td>
<td>Operator A has more knowledge about the environment.</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Realtime processing</td>
</tr>
<tr>
<td>Sequence:</td>
<td><strong>Main action sequence</strong></td>
</tr>
<tr>
<td></td>
<td>1. Operator A is finished with (a part of) his task, and has some spare time.</td>
</tr>
<tr>
<td></td>
<td>2. Operator A clicks on the messages tab on the SmartOps system, and reads the information that is shown on the screen.</td>
</tr>
<tr>
<td></td>
<td>3. Operator A continues with his tasks, or starts a new one.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UC 2</th>
<th>Anticipation and capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>...</td>
</tr>
<tr>
<td>Goals:</td>
<td>...</td>
</tr>
<tr>
<td>Description:</td>
<td>...</td>
</tr>
<tr>
<td>Pre-Conditions:</td>
<td>...</td>
</tr>
<tr>
<td>Post-Conditions:</td>
<td>...</td>
</tr>
<tr>
<td>Requirements:</td>
<td>...</td>
</tr>
<tr>
<td>Sequence:</td>
<td><strong>Main action sequence</strong></td>
</tr>
<tr>
<td></td>
<td>1. <em>This use case was left out since it contains confidential information.</em></td>
</tr>
</tbody>
</table>
### UC 3 Message Acknowledge

**Actors:** Sender A and directly addressed receiver B.

**Goals:** Inform the sender of the message that his message was read.

**Description:** An operator receives a message that was directed to him, and sends an acknowledge message to the sender. Only receivers that the sender has given up as receiver can acknowledge a message.

**Pre-Conditions:** A has information that is relevant for B.

**Post-Conditions:** B knows the information A sent. A knows B has read the information.

**Requirements:** Realtime processing, Acknowledgement possibility, Reply function, Messaging, Message Topics, Roger function

**Sequence:** **Main action sequence**

1. A sends a message with information to B.
2. B receives the message and wants to acknowledge he read the message.
3. B clicks on the "Roger" button of the text message.
4. It is shown to A that B had read the information.

### UC 4 Increase amount of info

**Actors:** Operator A.

**Goals:** Read more information and build up a better situational awareness.

**Description:** By utilizing the slider bar on the SmartOps system the operator can view more less relevant messages.

**Pre-Conditions:** Operator A has some spare time, not much information is shown on his screen.

**Post-Conditions:** More less-relevant information is shown on the screen.

**Requirements:** Filter information, Modify relevance filter

**Sequence:** **Main action sequence**

1. Operator A looks at the message screen and sees a few messages.
2. Operator A changes his relevance slider.
3. More less-relevant information appears on the screen.
4. Operator A reads the information.
5. Operator A has more knowledge about the environment and continues his tasks.
<table>
<thead>
<tr>
<th>UC 5</th>
<th>Reduce amount of info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Operator A</td>
</tr>
<tr>
<td>Goals:</td>
<td>Show only relevant information on the screen, so the operator has more time to complete his tasks.</td>
</tr>
<tr>
<td>Description:</td>
<td>Because an operator has to do a lot of tasks, he wants to have less information on his message screen.</td>
</tr>
<tr>
<td>Pre-Conditions:</td>
<td>Operator A has too many tasks (too high workload). A lot of information is shown on his message screen.</td>
</tr>
<tr>
<td>Post-Conditions:</td>
<td>Only relevant information is shown, so operator A has enough time to complete his tasks, but still has enough information about the environment.</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Filter information, Modify relevance filter</td>
</tr>
<tr>
<td>Sequence:</td>
<td><strong>Main action sequence</strong></td>
</tr>
<tr>
<td></td>
<td>1 Operator A switches to information view, and sees a lot of information on his screen.</td>
</tr>
<tr>
<td></td>
<td>2 Operator A changes his relevance slider.</td>
</tr>
<tr>
<td></td>
<td>3 Less relevant information disappears from the screen.</td>
</tr>
<tr>
<td></td>
<td>4 Operator A reads the relevant information, to get a quick situational awareness.</td>
</tr>
<tr>
<td></td>
<td>5 Operator A continues his task(s).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UC 6</th>
<th>Message Reply</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Operator A.</td>
</tr>
<tr>
<td>Goals:</td>
<td>An operator wants to reply to a message he received.</td>
</tr>
<tr>
<td>Description:</td>
<td>A user sends a reply on a received message.</td>
</tr>
<tr>
<td>Pre-Conditions:</td>
<td>Operator A has received a message. Operator A would like to reply to the message.</td>
</tr>
<tr>
<td>Post-Conditions:</td>
<td>Operator A has replied to the message. The system has calculated if the message is relevant for any other users.</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Reply function, Messaging, Message Topics</td>
</tr>
<tr>
<td>Sequence:</td>
<td><strong>Main action sequence</strong></td>
</tr>
<tr>
<td></td>
<td>1 Operator A indicates he would like to reply to a specific message.</td>
</tr>
<tr>
<td></td>
<td>2 The system allows the Operator A to enter a reply.</td>
</tr>
<tr>
<td></td>
<td>3 Operator A enters a reply and sends it.</td>
</tr>
<tr>
<td></td>
<td>4 The system calculates if the message is relevant for any other user.</td>
</tr>
</tbody>
</table>
**UC 7**  View specific discussion

**Actors:** Operator A, discussing groups.

**Goals:** View only one specific discussion.

**Description:** The operator is interested in one specific discussion between others and wants to view this discussion.

**Pre-Conditions:** Multiple discussion are being held.

**Post-Conditions:** One specific discussion is displayed.

**Requirements:** Realtime processing, Filter information, Personal messages

**Sequence:**

1. Operator A indicates he wishes to view only one specific discussion.
2. Operator A selects discussion
3. The system only displays one specific discussion.

**UC 8**  View all relevant messages

**Actors:** Operator A.

**Goals:** Having an overview of all discussions.

**Description:** The user wants to view all relevant messages from every discussions.

**Pre-Conditions:** The system only displays one specific discussion.

**Post-Conditions:** The user sees all relevant messages from all discussions.

**Requirements:** Filter information

**Sequence:**

1. Operator A indicates he wishes to view all discussions.
2. Operator A selects relevance he wishes to use as filter
3. The system shows all relevant messages from all discussions.
### UC 9  
**Unaddressed message sending**

**Actors:** Operator A.

**Goals:** Inform other users.

**Description:** The user wants to send an unaddressed message, in order to inform potential interested others.

**Pre-Conditions:**

**Post-Conditions:** The entered message has been send to the SmartOps System. Sent message can be seen by Operator A.

**Requirements:** Messaging

**Sequence:**  
**Main action sequence**

1. Operator A indicates he wishes to send a message.
2. System allows operator A to enter a message.
3. Operator A enters a message.
4. Operator A submits the entered message.
5. Operator A sees his submitted message.
6. Operator A sees who has read his message.

### UC 10  
**Addressed message sending**

**Actors:** Operator A.

**Goals:** The user wants to give information to another user.

**Description:** The user wishes to send a message to a specific person. Extends: Unaddressed message sending.

**Pre-Conditions:**

**Post-Conditions:** The message has been send to the addressed person. Based on relevance and other settings, other users may also be able to see the message.

**Requirements:** Messaging, Personal messages

**Sequence:**  
**Main action sequence**

3b. Operator A specifies to whom the message is directly addressed.
<table>
<thead>
<tr>
<th>UC 11</th>
<th>Search for a Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Admiral A.</td>
</tr>
<tr>
<td>Goals:</td>
<td>Finding a previously seen message.</td>
</tr>
<tr>
<td>Description:</td>
<td>A user has entered/read a message some time ago, and he or she wants to re-read that message again.</td>
</tr>
<tr>
<td>Pre-Conditions:</td>
<td>Admiral A has sent a message in SmartOps. Admiral A wants to read that message again, but can not see it on his screen.</td>
</tr>
<tr>
<td>Post-Conditions:</td>
<td>Admiral has seen the message.</td>
</tr>
<tr>
<td>Requirements:</td>
<td>Messaging, Search functionality</td>
</tr>
</tbody>
</table>
| Sequence: | **Main action sequence**

1. Admiral A needs a previously seen message for his task, but he does not see it on his screen.
2. Admiral A indicates to the system he wishes to search.
3. Admiral A enters a keyword for the message he is looking for.
4. Admiral A optionally indicates whether he wishes to see sent messages, received messages or both.
5. The system shows a few messages.
6. Admiral A reads the message he searched for.
7. Admiral A continues his task. |

<table>
<thead>
<tr>
<th>UC 12</th>
<th>Use touchscreen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Operator A.</td>
</tr>
<tr>
<td>Goals:</td>
<td>Faster interaction or better ergonomics.</td>
</tr>
<tr>
<td>Description:</td>
<td>The user wants interact with the system using the touchscreen interface</td>
</tr>
<tr>
<td>Pre-Conditions:</td>
<td></td>
</tr>
<tr>
<td>Post-Conditions:</td>
<td></td>
</tr>
<tr>
<td>Requirements:</td>
<td>Touch compatible</td>
</tr>
</tbody>
</table>
| Sequence: | **Main action sequence**

1. The system always allows the user to interact using the touchscreen.
2. Operator A (temporarily) interacts with the system using the touchscreen. |
### UC 13: Log playback

**Actors:** Operator B.R. Am

**Goals:** The goal is to gain insight into the results of the system, so the relevance determination can be improved.

**Description:** Wilhelmus wants to load and playback a scenario log with messages.

**Pre-Conditions:**

**Post-Conditions:** Scenario has been played back.

**Requirements:** Log playback should be possible

**Sequence:**

1. The operator indicates he wishes to load and play back the log.

### UC 14: Changing message relevance

**Actors:** Operator A

**Goals:** The system will be able to better estimate the relevance of future messages and the user provides ordering for himself.

**Description:** The user changes the relevance of a message.

**Pre-Conditions:** Operator A has received a message

**Post-Conditions:** The relevance of future message will be different. The ordering of the message itself has been changed.

**Requirements:** Modify relevance filter, Relevance based on user feedback

**Sequence:**

1. Operator A changes the relevance of a message.
2. The system changes the ordering of the message.

### UC 15: Expert changing rules.

**Actors:** Expert X.

**Goals:** There is a set of fuzzy rules which can be used by the system to score a message on relevance.

**Description:** The expert translates his rules to a set of fuzzy rules which can be used by a fuzzy expert system.

**Pre-Conditions:** The expert has in depth knowledge of the problem in the field.

**Post-Conditions:** The expert has translated a solution for the problem into fuzzy rules, in such a way a fuzzy expert system can use it.

**Requirements:** Relevance based on fuzzy expert system

**Sequence:**

1. The expert starts a text editor, and loads a set of fuzzy rules.
2. The expert adds, removes and alter some fuzzy rules.
3. The expert saves the text file and exits the editor.
### Appendix B

#### Requirements

<table>
<thead>
<tr>
<th>Requirement 1</th>
<th>Realtime processing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>The system shall process information real time.</td>
</tr>
<tr>
<td><strong>Claim:</strong></td>
<td>Up-to-date SA</td>
</tr>
<tr>
<td></td>
<td><em>Real time processing allows the situational awareness for the user to be continuously up-to-date.</em></td>
</tr>
<tr>
<td></td>
<td>+ User will respond quicker.</td>
</tr>
<tr>
<td></td>
<td>− The user can be overwhelmed with information if no filtering takes place.</td>
</tr>
<tr>
<td><strong>Related Use Cases:</strong></td>
<td>Reading Information, Anticipation and capabilities, Message Acknowledge, View specific discussion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement 2</th>
<th>Filter information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description:</strong></td>
<td>The system shall only display relevant information for a specific actor.</td>
</tr>
<tr>
<td><strong>Claim:</strong></td>
<td>Workload decreases</td>
</tr>
<tr>
<td></td>
<td><em>Because only relevant information is showed, the workload of the operator is expected to decrease.</em></td>
</tr>
<tr>
<td></td>
<td>+ Less time is required to view the information.</td>
</tr>
<tr>
<td></td>
<td>+ Less time is needed to determine whether information is relevant or not.</td>
</tr>
<tr>
<td></td>
<td>+ Less time required to build situational awareness.</td>
</tr>
<tr>
<td></td>
<td>− The operator may miss relevant information because it is classified incorrectly.</td>
</tr>
<tr>
<td><strong>Claim:</strong></td>
<td>Anticipation Increases</td>
</tr>
<tr>
<td></td>
<td><em>Because relevant information is showed to the operator, a better situational awareness is gained.</em></td>
</tr>
<tr>
<td></td>
<td>+ The operator is able to anticipate on future events.</td>
</tr>
<tr>
<td></td>
<td>+ If the operator has spare time, he can read more information and do tasks in advance, to spread the workload.</td>
</tr>
<tr>
<td></td>
<td>− The operator might get too much information if there is happening a lot.</td>
</tr>
<tr>
<td></td>
<td>− The operator might get too much information because it is classified incorrectly.</td>
</tr>
<tr>
<td><strong>Related Use Cases:</strong></td>
<td>Anticipation and capabilities, Increase amount of info, Reduce amount of info, View specific discussion, View all relevant messages</td>
</tr>
<tr>
<td>Requirement 3</td>
<td>Modify relevance filter</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Description:</td>
<td>It should be possible to modify the sensitivity of the relevance filter.</td>
</tr>
</tbody>
</table>
| Claim: | Better work environment  
*The operator can set his or her own amount of messages based on a filter threshold to act on.*  
+ The operator does not see too many or too few messages.  
− The operator has to set the filter by him- or herself. |
| Related Use Cases: | Increase amount of info, Reduce amount of info, Changing message relevance |

<table>
<thead>
<tr>
<th>Requirement 4</th>
<th>Acknowledgement possibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The system should allow the user to give acknowledgements of received messages.</td>
</tr>
</tbody>
</table>
| Claim: | Clear acknowledgements  
*By adding an acknowledgement option to text messages it can be made clear to the sender the message has been received and read.*  
+ Sender knows the message has been read, which also expands his environmental awareness.  
+ The directly addressed receiver knows the sender can see he has read the message.  
+ The system knows which acknowledgements are related to other messages, so relevance can be better determined.  
− It takes a little more work to press the acknowledge button.  
− It takes some discipline to use this function each time. |
| Related Use Cases: | Message Acknowledge |

<table>
<thead>
<tr>
<th>Requirement 5</th>
<th>Reply function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The system could allow the user to indicate whether a message is a reply and to which message it replies.</td>
</tr>
</tbody>
</table>
| Claim: | Thread indication  
*The system can always indicate to which thread a message belongs.*  
+ Because of the replies users can easily see responses on a message.  
+ It can be easier to determine relevance, because replies link messages together.  
− Long threads of replies might occur, which might become chaotic.  
− Determination of relevance can be wrong if people reply but change the topic.  
− It requires the user to use the reply function (in the right way), which requires time. |
<p>| Related Use Cases: | Message Acknowledge, Message Reply |</p>
<table>
<thead>
<tr>
<th>Requirement 6</th>
<th>Messaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>Users should be able to create a message, and should be able to read messages.</td>
</tr>
</tbody>
</table>
| Claim: | Communication  
*Users communicate in such a way that they can inform others.*  
+ Users can anticipate to others.  
− When receiving a message, users may be interrupted when doing their current task. |
| Related Use Cases: | Anticipation and capabilities, Message Acknowledge, Message Reply, Unaddressed message sending, Addressed message sending, Search for a Message |

<table>
<thead>
<tr>
<th>Requirement 7</th>
<th>Search functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The system could allow the user to search through his or her messages in order to find one or more specific message.</td>
</tr>
</tbody>
</table>
| Claim: | Faster working  
*The user has only messages of interest on his screen, so he does not lose time searching them.*  
+ The user spends less time searching all messages.  
− The user has to think of a correct search term. |
| Related Use Cases: | Search for a Message |

<table>
<thead>
<tr>
<th>Requirement 8</th>
<th>Touch compatible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The system could allow the user to interact with the system by touching the screen instead of the conventional mouse and keyboard.</td>
</tr>
</tbody>
</table>
| Claim: | User can work faster  
*By using a touch screen the user uses less time as he can use both hands.*  
+ The user has more time for other tasks.  
+ The user can work quicker.  
− The use of a touchscreen is less ergonomic. |
| Related Use Cases: | Use touchscreen |

<table>
<thead>
<tr>
<th>Requirement 9</th>
<th>Log playback should be possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The system shall be able to playback a recorded log file with messages and process it.</td>
</tr>
</tbody>
</table>
| Claim: | Testing allows adjustments  
*Testing on scenarios allows the system to be adjusted to the real world*  
+ System behaves better to the real world.  
− System may react too specific to one single test case. |
<p>| Related Use Cases: | Log playback |</p>
<table>
<thead>
<tr>
<th>Requirement 10</th>
<th>Personal messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>It should be possible to send personal messages to a person.</td>
</tr>
<tr>
<td>Claim: Personal messages</td>
<td>The user can work better if he can differentiate between personal and non-personal messages, as indicated by the sender. + The receiver can more quickly view what messages are directed at him. + It is easier to determine the relevance of all messages, if the system can use the information of what messages are personal. – It costs more time for the sender to add a receiver to the message, to indicate to whom the message is directly addressed to.</td>
</tr>
<tr>
<td>Related Use Cases:</td>
<td>View specific discussion, Addressed message sending</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement 11</th>
<th>Message Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The system should always be able to indicate to what topic messages belong.</td>
</tr>
<tr>
<td>Claim: Clear topics</td>
<td>It should be clear to what topic messages belong. + Because of the replies users can easily see responses on a message. + It can be easier to determine relevance, because replies link messages together. – Long threads of replies might occur, which might become chaotic. – Determination of relevance can be wrong if people reply but change the topic.</td>
</tr>
<tr>
<td>Related Use Cases:</td>
<td>Message Acknowledge, Message Reply</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement 12</th>
<th>Roger function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The system could allow the user to indicate he has rogered a message and also display to the sender what message was rogered and by whom.</td>
</tr>
<tr>
<td>Claim: Roger indication</td>
<td>The system can always indicate to whether a message has been rogered. + The user can always see what message was rogered by whom. – The user is required to use the specific roger function.</td>
</tr>
<tr>
<td>Related Use Cases:</td>
<td>Message Acknowledge</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement 13</th>
<th>Relevance based on user feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The system could use the relevance determination of previous messages as determined by the user, to judge the relevance of new messages.</td>
</tr>
<tr>
<td>Claim: Relevance of user feedback</td>
<td>The system will be able to better judge the relevance of incoming messages, based on the relevance determination given by the user of previous messages. + The system will focus more on the subjects the user finds relevant. – If a subject was found to be irrelevant, but suddenly has become relevant because of a developing situation, the first message should actually be judged as more relevant. When this first message is not displayed at all, this could stay unnoticed.</td>
</tr>
<tr>
<td>Related Use Cases:</td>
<td>Changing message relevance</td>
</tr>
<tr>
<td>Requirement 14</td>
<td>Relevance based on fuzzy expert system</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>The system shall use a set of fuzzy rules and a fuzzy expert system to determine the relevance score for a message. The fuzzy expert shall use the metadata of the operator for whom the relevance is determined, such as the receiver’s location. The fuzzy expert system also shall use meta data from the message, such as its sent location or content. Additionally the fuzzy expert system uses relation information between sender and the operator for whom the relevance is determined, such as social distance between those two.</td>
</tr>
<tr>
<td><strong>Claim:</strong></td>
<td>Easy Maintainability</td>
</tr>
<tr>
<td></td>
<td><em>Using a fuzzy expert system an expert on the field can easily change rules.</em></td>
</tr>
<tr>
<td></td>
<td>+ Adding and removing rules can be done easily by an expert in the field.</td>
</tr>
<tr>
<td></td>
<td>+ Effects of a rule can easily be understood.</td>
</tr>
<tr>
<td></td>
<td>+ Can deal with fuzzy or incomplete data.</td>
</tr>
<tr>
<td></td>
<td>– Rules have to be created correctly.</td>
</tr>
<tr>
<td></td>
<td>– The expert should understand how fuzzy rules work.</td>
</tr>
<tr>
<td><strong>Related Use Cases:</strong></td>
<td>Expert changing rules.</td>
</tr>
</tbody>
</table>
Appendix C

Graphical User Interface Designs

Several Graphical User Interface designs were made. The designs differ in how information is displayed and used to lay-out user interface components.

For example, each message has a relevance score. This relevance score can be displayed to the user by giving the message a colour based on this relevance. Another option is to position messages differently based on relevance. Whether a message is addressed to the user, is also information. This can also be used to give different colours based on whether the message is addressed to the user.

All these factors allow many different graphical user interface designs. The first step was to come up with a small set of graphical user interface designs, based on our own opinion on what might work.

C.1 Two column layout

The first design reflects the basic concept of this project: when an operator has extra time, they are able to read the most relevant messages. The design is conceptually sketched in Figure C.1.

![Figure C.1: First design.](image)

The proposed design consists of two columns. The left column list all messages which are addressed to the user. These messages are the messages the receiver would receive in the current situation. The messages are ordered by time.

The right column are the relevant messages for the operator, which were not addressed to the user. These messages are ordered by relevance. This column will be scanned by the user if the user has additional time. These messages, ordered by relevance is the extra information the system will display, compared to the current situation.

C.2 Enhanced two column layout

In the first design, when the user has additional time, the user can simply scan the right column from top to bottom. The most relevant messages will then be read. A disadvantage of this design is that when
the user scans the relevant messages, the context is no longer clear. If a message can only be understood with the previous messages, the ordering is unsuitable.

The requirement would then be to have relevant messages be both ordered by relevance and time. This can be done by ordering messages by time on the Y-axis and by relevance of the X-axis. Relevant messages will always be displayed on the left of the column, less relevant messages to the right of the column. This design is conceptually sketched in Figure C.2

![Figure C.2: Second design.](image)

C.3 Integrated design

The advantage of the previous two column layouts is that the user always knows that the right column never displays directly addressed messages, so this screen column should not have to be constantly watched. However, when a relevant message is read and has a relation with messages in the left column, the context becomes scattered between these two columns.

The solution is to integrate the two columns. Again, more relevant messages will be displayed more to the left and the Y-axis is used for time ordering. This design always shows a clear context. This design is conceptually sketched in Figure C.3

![Figure C.3: Third design.](image)

C.4 Integrated design with relevance slider

A disadvantage of the integrated design is that the user has to scroll much, since all messages are listed in one column. The user may lose the total picture. When the operator has a high workload, he may only want to see the messages addressed to him. If the workload is low and the operator has additional time, he may be interested in both addressed messages, but also messages of high and medium relevance.

A solution is to add a slider on top of the screen. This slider can then be adjusted by the user to determine how many messages he wants to see. All message to the right of the slider will no longer
be displayed, and the list of messages will be shorter. If the slider is set all the way to the left, only the messages which were addressed to the user will be displayed. This allows the user to adapt the number of displayed messages on his workload. This design is conceptually sketched in Figure C.4

![Figure C.4: Fourth design.](image)

C.5 Threaded view

A list of messages can be read much faster if it is one conversation, instead of a list of messages were multiple conversations are interweaved. It would therefore be an advantage if messages are grouped by conversation or thread. Multiple threads can be displayed in multiple columns. This design is conceptually sketched in Figure C.5

Relevance of messages can be indicated with colours. A red colour could be used for relevant messages, while less relevant messages are green. Transparency could also be used; relevant messages can be made opaque, while less relevant messages are displayed increasingly more transparent.

This design also has some disadvantages. The first problem is that for each message, the thread to which it belongs should be known. This could be determined automatically, or by a human. The sender could indicate the thread to which the message belongs using reply buttons, or a letter code. This requires some etiquette and it begs the question whether this etiquette will be followed during high workloads. It should be noted that thread determination is also most important during high workload. Thread determination could also be done automatically, using LSI. A hybrid system in which both the system itself and the sender determine the correct thread is also imaginable.

Another problem is that threads are not always defined clearly and are not static. New threads can develop, threads can merge, threads can belong to other threads, and one message can belong to multiple threads. How these problems can be taken into account in this design is complex.

![Figure C.5: Fifth design.](image)
C.6 Dynamic threads

Some kind of thread indication is an advantage, but the previously stated threaded view has some disadvantages.

A solution could be to take the integrated design with relevance slider and add thread highlighting. This design is conceptually sketched in Figure C.6. When the user selects a message, all messages from this thread are highlighted. With a different action, say a double click, the user is able to only display this thread. Moreover, thread determination can be dynamic. When the system needs to highlight the thread of a message, it can use LSI to highlight the messages which are similar to the selected message. It no longer is a problem if messages belong to multiple threads.

![Diagram](image)

Figure C.6: Sixth design.
Appendix D

Interview with experts about GUI proposals

In this section we describe our conclusions about the opinions of one officer and two noncommissioned officers we interviewed about the presentation of information. We presented some lay-out examples we thought of our own and enabled a conversation about these lay-outs. The lay-outs were presented on paper, with separate pieces of paper for each message, since this allows good explanation. An example presentation can be seen in Figure D.1. With this interview we wanted to get to know what is practical in the user interface and what needs the operators have.

Figure D.1: Items used for the presentation of lay-outs.

From the interview with the first operator we concluded that he did not want a lot of different windows on his screen. He did not want to do a lot of actions in order to view the messages. He preferred to
have all the text messages together. To show the different types of information (personal or determined by relevance) he preferred to use highlighting. He mentioned highlighting based on topic (by selecting a message) could be very handy. He also would like to have multiple (also non-chat, like the internet browser) windows open at the same screen.

The second operator we interviewed, cared most about having not too much information on the screen. He preferred a single box with information above having multiple threads. Having multiple threads would reduce the clarity of the screen in his opinion. He also expected that by letting the user chose if more or less information should be shown, users would not chose to see much information.

The last operator wanted to be sure he missed no information. He preferred to see more information on his screen than missing out on it. He also mentioned it should be possible to view messages in short moment of time. For this it should be clear which messages have a high relevance or not. The operator suggested to use highlighting for this. He was not enthusiastic to show the relevance on the X-axis, because it reduces the clarity. Then it would not be possible any more to view messages in a short moment of time.

He also was positive about splitting up the messages into multiple threads, based on their topics. His suggestion was letting the sender of the message specify predetermined keywords to messages, in order to make the topics more clear.

We came to the conclusion the preferences for the user interface are very personal. The idea behind the user interface is the same, the information should be shown in a clear way, and it is useful to be able to see to what topic a message belongs. About how this should be done all operators had their own preference. While the second operator liked to have all messages together, the third operator preferred to have multiple threads with a topic for each thread. They all were positive about highlighting the topic of messages when for example clicking on a message.
Appendix E

Test Plan

E.1 Introduction

Software can be tested in many ways and both thoroughly and superficially. How testing should be carried out depends on the requirements on the software’s maturity.

This goal of this project is not to deliver mature software which can be used in the field. Instead, a proof of concept will be made. The maturity of software can be defined in several ways. One way is to use the Software Technology Readiness Level (TRL) as defined by the United States Department of Defence (DoD). The software built will have a software TRL of level 3. The DoD defines a TRL of level 3 as follows: “Analytical and experimental critical function and/or characteristic proof of concept. [...] Active R&D is initiated. The level at which scientific feasibility is demonstrated through analytical and laboratory studies. This level extends to the development of limited functionality environments to validate critical properties and analytical predictions using non-integrated software components and partially representative data.” [30]

Since the software will be a proof of concept, the test plan should be made keeping this in mind. This means the focus will be on testing the claims as defined in Appendix H.

E.2 Functional Testing

E.2.1 Automated unit testing

As Edsger W. Dijkstra stated: “Testing shows the presence, not the absence of bugs.” Testing can almost never guarantee that an application is bug free, but can certainly help to improve software quality.

However, thorough automated unit testing requires a lot of time. Both positive and negative tests need to be made. Moreover, infinite different test cases are possible and test cases can be combined in endless ways. Therefore, to assure an acceptable level of software quality, a representative set of test cases should be made. This set is still large. Moreover, all these test cases need to be documented and maintained. Unit testing therefore requires a lot of time.

Automated unit testing is also helpful in regression testing. Regression testing is most important when new software updates are made. Since only one version will be made of the developed software, the need for regression testing is low.

Because of our restricted time and since the project will have a TRL of level 3, it was chosen to not carry out automated unit testing. Other testing methods will be used as described in the next paragraphs.

E.2.2 Assertions

Assertions are boolean expressions in software which should evaluate to true. At unit level, assertions will be used to guarantee pre- and post-conditions of some methods.
E.2.3 Module testing

All system components will use our own developed database library. The need for this module to function as specified is somewhat higher than for other modules. Therefore, some basic tests will be made for this module, to test whether the interface functions according to specification.

During the development of the database library, the other components will be tested by temporarily simulating the database library. This will be done by simply calling the right methods as if the database library would have done this.

E.2.4 Integration Testing

As described in the Architectural Design Document (ADD), all software applications will interface via the database. All software applications will use our own database layer. Between the software applications and this developer library the largest implementation testing will take place.

The Architectural and Technical Design were of course developed with this interface in mind. The integration test will take place when all software applications run. When all applications use the data from the database produced by other applications in the right way, the test will be considered successful. During such a test, the interface with the database layer and the software application is tested, the interface with the database library and the database itself and the interaction between the different software applications via the database.

This a lot to test in on time, therefore the database library will be tested by hand with the database. In this test, only a few functions will be tested.

As described in the ADD, not all components will be implemented. Therefore, only integration tests will be carried out for the implemented components: the ticker, message parser, relevance algorithm and user application. Keeping the section Component Communication in mind, integration cases can be formulated.

In the first integration test, the message parser will be run. Messages will be inserted in the database, which will be detected by the relevance algorithm. The relevance algorithm will put the relevance of these messages in the database. The user application will detect this and display the messages with relevance.

In the second integration test, the ticker will be run, with messages already stored in the database with relevance, being displayed by the user application. The ticker should then update the age of the messages, after which the relevance algorithm updates the relevance of messages, which should be seen in the user application.

E.2.5 Stability Testing

The created software should be stable, meaning that it should not crash. The stability of the application will be tested by running the application for an acceptable period. In this period one scenario should be completely executed without crashing. In our case we will run the Noordzee scenario provided by TNO. If the scenario runs without crashing the stability test has been successful.

E.2.6 System Testing

A pilot is scheduled to see if we can quantify our claims. During the pilot a questionnaire was used to make sure that all the claims would be tested Table E.1 links the questions to the claims they test. The questionnaire with the answers from the expert can be found in Appendix F. The pilot will consist of eleven information moments where the simulation will be stopped and questions will be asked to the user. After scenario a final round of questions will be asked.

E.3 Testing claims
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Implemented</th>
<th>Claim</th>
<th>Sub claim</th>
<th>Claim number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Up-to-date SA</td>
<td>User will respond quicker.</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The user can be overwhelmed with information if no filtering takes place</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Less time is required to view the information.</td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Less time is needed to determine whether information is relevant or not.</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Less time required to build situational awareness.</td>
<td>2.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator may miss relevant information because it is classified wrong.</td>
<td>2.4</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>Workload decreases</td>
<td>The operator is able to anticipate on future events.</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the operator has spare time, he can read more information and do...</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator might get too much information if there is happening a lot.</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator might get too much information because it is classified wrong.</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator does not see too much or too less messages.</td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator has to set the filter by him- or herself.</td>
<td>3.2</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>Better work environment</td>
<td>The operator is able to anticipate on future events.</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the operator has spare time, he can read more information and do...</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator might get too much information if there is happening a lot.</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator might get too much information because it is classified wrong.</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator does not see too much or too less messages.</td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator has to set the filter by him- or herself.</td>
<td>3.2</td>
</tr>
<tr>
<td>4</td>
<td>No</td>
<td>Clear acknowledgements</td>
<td>The operator is able to anticipate on future events.</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the operator has spare time, he can read more information and do...</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator might get too much information if there is happening a lot.</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator might get too much information because it is classified wrong.</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator does not see too much or too less messages.</td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The operator has to set the filter by him- or herself.</td>
<td>3.2</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Thread indication</td>
<td>Users can anticipate to others.</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When receiving a message, users may be interrupted when doing...</td>
<td>6.2</td>
</tr>
<tr>
<td>6</td>
<td>Yes</td>
<td>Communication</td>
<td>Users can anticipate to others.</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When receiving a message, users may be interrupted when doing...</td>
<td>6.2</td>
</tr>
<tr>
<td>7</td>
<td>No</td>
<td>Faster working</td>
<td>Users can anticipate to others.</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When receiving a message, users may be interrupted when doing...</td>
<td>6.2</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>User can work faster</td>
<td>Users can anticipate to others.</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When receiving a message, users may be interrupted when doing...</td>
<td>6.2</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Testing allows adjustments</td>
<td>Users can anticipate to others.</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When receiving a message, users may be interrupted when doing...</td>
<td>6.2</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Personal messages</td>
<td>Users can anticipate to others.</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When receiving a message, users may be interrupted when doing...</td>
<td>6.2</td>
</tr>
<tr>
<td>11</td>
<td>No</td>
<td>Clear topics</td>
<td>Users can anticipate to others.</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When receiving a message, users may be interrupted when doing...</td>
<td>6.2</td>
</tr>
<tr>
<td>12</td>
<td>No</td>
<td>Roger indication</td>
<td>Users can anticipate to others.</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>When receiving a message, users may be interrupted when doing...</td>
<td>6.2</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>Relevance of user feedback</td>
<td>The system will focus more on the subjects the user finds relevant.</td>
<td>13.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If a subject was found to be irrelevant, but suddenly has become...</td>
<td>13.2</td>
</tr>
<tr>
<td>14</td>
<td>No</td>
<td>Easy rules maintainability</td>
<td>The system will focus more on the subjects the user finds relevant.</td>
<td>13.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If a subject was found to be irrelevant, but suddenly has become...</td>
<td>13.2</td>
</tr>
</tbody>
</table>

Table E.1: Table on how to test the claims.
Appendix F

Pilot Test Results

This chapter contains the questions asked during the pilot and answers received by the expert. Besides the scores the expert gave also all remarks are added to the questions. The expert could score questions on a scale from 1 to 10, a 1 indicates the expert agrees and a 10 that he disagrees with the question.

During the scenario:

1. At this moment, do you feel that you have a better understanding of the situation because of the relevance system?
   - Claim: 2.3
   - Score: 3/10
   - Notes: The user could see how the system would aid him in better understanding the situation. However the messages in the scenario were for the most part not relevant because they would already be communicated vocally. The user indicated that a potential other use for the application is that when a HDO is going to start his shift he has the ability to read the most relevant messages and get a good idea about the situation without being in the command center.

2. At this moment, do you feel you can make a more accurate decision with the extra information provided by the relevance system?
   - Claim: 1.1
   - Score: 2/10
   - Notes: The user referenced to the points he made at the first question.

3. At this moment, do you think you can quicker anticipate on future events?
   - Claim: 2.5
   - Score 2/10
   - Notes: The user referenced to the points he made at the first question.

4. At this moment, do you see any messages that you would like to give another relevance score than the system gave it?
   - Claim: 2.4
   - Claim: 13.1
   - Notes: The user noted that certain messages like messages about the planning should have a long lasting relevance till they are read.

After the scenario:

1. Do you think as a HDO you have enough time to use the relevance system in between your normal work?
2. Will the relevance system help you do your work more efficiently?

- Claim: 2.1
- Claim: 2.2
- Score: 2/10
- Notes:

3. Do you think that there is a negative impact by receiving more messages and causing more interrupts to the user's normal work?

- Claim: 6.1
- Claim: 6.2
- Score: 2/10
- Notes:

4. Does the slider give you enough control to adjust the amount of information that is displayed?

- Claim: 1.2
- Claim: 2.7
- Claim: 3.1
- Score: 2/10
- Notes: The user noted that it’s very important have enough trust in the system to optimally use the slider and not be afraid to miss some information.

5. Do you think adjusting the slider would cost a lot of your time?

- Claim: 3.2
- Score: 7/10
- Notes: The user thinks it will take some time to get used to the slider but will in the long run will yield more time.

6. Would you like an automated slider better? (The system will try to get an indication about how busy you are at the moment and increase or decrease the amount of messages displayed)

- Claim: 3.2
- Score: 9/10
- Notes: The user thinks it will be very hard to get an indication how busy he is at a moment as you get trained to multitask extremely effective.

7. Do you feel that having the ability to mark messages as read makes you able to work more efficiently with the messages from the relevance system?

- Claim: 6.1
- Score: 1/10
- Notes: By combining this and the slider the user can easily keep his message list short and not be drawn to the screen by a large amount of messages.
Appendix G

Fuzzy Variables

This appendix was left out.
Appendix H

Fuzzy rules not time-based

This appendix was left out.
Appendix I

Fuzzy rules time-based

This appendix was left out.
Appendix J

User Manual

Note

In this document *MsgRelDet* refers to the software package you are about to install and run.

J.1 Introduction

This document describes how *MsgRelDet* can be installed, and executed. As stated in the *MsgRelDet* documentation the software consists of several separate software programs. These components make use of one single database. In order to run *MsgRelDet* the database has to be set up first. Next some, or all, applications need to be started in a specific order. This document describes how to install the database and how to start the software.

J.2 System requirements

**Trex:** Windows Vista or higher.

**UserApp, Relevance Determination Algorithm, and Ticker:** A computer with Java Runtime Environment\(^1\) (or Java) installed.

J.3 Setting up the database

*MsgRelDet* needs a Sesame database for storing data. The Sesame database application can be downloaded from its website at [http://www.openrdf.org/download.jsp](http://www.openrdf.org/download.jsp). If *MsgRelDet* is plugged in to another Mutual Empowerment Prototype, the database setup and configuring part can be skipped.

J.3.1 Installing the database

The installation of the database can be done in several ways. In this manual two possibilities are given.

- The recommended way by the Sesame website, is to install Sesame in an Apache Tomcat environment. The Mutual Empowerment Prototype Development Kit document ([31]) provides a how-to do this.
- The Sesame database can be executed from a Jetty environment. No installation is needed. *MsgRelDet* provides a generic Java application to start the database without installing it to the computer system. The following steps describe how to start the database in this way.
  1. Go to the folder
  2. Double click Start.jar

J.3.2 Configuring the database

Once the database is installed and running, a new repository has to be created. MsgRelDet will store its data in that repository. The following steps describe how a repository can be created.

1. Start your browser and go to the following URI: http://localhost:8080/openrdf-workbench.
2. Click on “New repository”.
3. Enter an unique ID, for example “RelevanceData”.
4. Click on next.
5. Click on create.
6. You get an overview with some settings. The repository has been created successfully.

J.4 Configuration files

Each software component comes in an own folder, and each software component comes with its own configuration. This config can be found in the file ./conf/mainconfig.xml, which can be opened with any text editor. In this file there is an option to set the database. Most important is the configuration to the database, which can be found under the field <repository>. The chosen repository name has to be filled in under the field <name>. All other options will not discussed in this manual.

J.5 Adding roles to the database

First the database needs to be populated with some data before MsgRelDet can do their job. MsgRelDet requires that all roles are loaded in the database. The role information is stored in a csv file. The following steps describe how to put this csv file in the database using Trex.

1. Start Trex
2. Click on Simulation
3. Click on Edit Simulation
4. Click on add Simulated Sensor
5. Click on Browse
6. Normally the role csv file can be found at the relative path: ...\Trex\SensorData\Relevance
7. Click on Add Sensor
8. Click on Start
9. The roles will be pushed to the database within a second.

J.6 Starting the UserApp (optional)

Once the role information is stored in the database, the UserApp can be started. Initially the UserApp starts with no messages. Once the relevance algorithm is started in a next step, the UserApp will display messages which are rated for the user (role) found in the config file. The following steps describe how to start the UserApp.

1. Open the folder ./userapp.
2. (Double)click on the JAR, or run in a commandline:
   
   java -jar userapp.jar

   61
J.7 Starting the ticker

The ticker is a component which ensures the relevance algorithm gets notifications to update certain messages once some time has passed.

1. Open the folder ./ticker.
2. (Double)click on the JAR, or run in a commandline:
   ```
   java -jar ticker.jar
   ```

J.8 Starting the relevance determination algorithm

Next the relevance algorithm should be started. This algorithm will give a relevance score to each new message in the database, or when a message has changed. Note that the relevance algorithm only rates the messages for the user specified in the config file.

1. Open the folder ./msgrelevancedet.
2. (Double)click on the JAR, or run in a commandline:
   ```
   java -jar algorithm.jar
   ```

J.8.1 Changing the fuzzy rules

The fuzzy rules are defined in a language called ”Fuzzy Control Language”. An easy explanation to FCL can be found at [http://jfuzzylogic.sourceforge.net/html/fcl.html](http://jfuzzylogic.sourceforge.net/html/fcl.html). It is advised to be familiar with FCL when modifying the fuzzy rules. The fuzzy rule file itself can be found in the installation directory of the Algorithm.

To create a new ruleset, just create a new .fcl file inside the “rulesets” folder (which is inside the installation directory). Then the configuration should be modified so it uses the newly created fcl file. In the configuration folder there is a file called settings.xml. This xml file contains all operators the relevance is determined for, and the corresponding options. The options called “ruleset” and “rulesettimed” can be modified to the absolute url to the created rulesets.

To modify an existing ruleset file, just edit it in any texteditor. The variables that can be used in the ruleset are described in Appendix G. Please note that all variables that are available should be defined, also if they aren’t used.

J.9 Emulating a scenario (optional)

MsgRelDet can play scenarios in a special log format. The log is stored in a comma separated values file (.csv). A scenario can be simulated in two ways. The first one is by using Trex, the second is by invoking the csvparser included in the algorithm file (see Appendix J.8).

J.9.1 Emulating a scenario using Trex

Simulating a scenario with Trex is very straightforward and follows almost identical steps as putting roles in the database.

1. Start Trex
2. Click on Simulation
3. Click on Edit Simulation
4. Click on add Simulated Sensor
5. Click on Browse
6. An example csv file can be found at the following relative path: ...\Trex \SensorData \Relevance \HDOmessagesRelevanceExtras.csv

7. Click on Add Sensor

8. The speed multiplier must be kept on 1.

9. Click on Start

10. The scenario will now start and push messages to database at the appropriate times.

**J.9.2 Emulating a scenario using the algorithm program**

To simulate a scenario using the algorithm, the configuration of the Algorithm should be modified so it uses the csv file for the input messages. This settings can be found in the configuration folder of the Algorithm installation directory. In that directory there is a settings.xml file, which contains an option called "experimentcsv". The value of this option should be set to the absolute url to the csv file to use as input.

When the configuration of the Algorithm is done, the applications should be started by following the next steps:

1. Start the database as described earlier
2. Start the Algorithm (execute the Algorithm.jar)
3. Start the ticker (execute the Ticker.jar)
4. Start the UserApp (execute the UserApp.jar)

When these steps are performed the UserApp should start showing messages when they are generated, and showing them ordered by relevance.
Bibliography


[27] Oracle, “About the jfc and swing.” [http://download.oracle.com/javase/tutorial/uiswing/start/about.html](http://download.oracle.com/javase/tutorial/uiswing/start/about.html).


[31] M. van Zomeren, “Mutual empowerment prototype development kit (me-pdk).” This is an unversioned draft user manual for TrexCore. In order to retrieve this document, one may contact the authors at TNO Soesterberg.
Project Initiation Document

Bram Beernink
Arjen Goedegebure
Niels van Kaam
Remco van der Zon

Summary

1 Introduction

This document is the project initiation document for our TU Delft Bsc project. As a final project for the study computer science we will execute a project at TNO Soesterberg.

In this document we will first give a clear description of the project assignment and environment. Next we describe our approach for completing the project. We give a brief overview of the expected project organization and list requirements by TNO and TU Delft. The project planning is described in the following section. Finally we discuss on how to ensure the quality of the project.

2 Project Assignment

Our project is part of the larger “SmartOps” project by TNO. The SmartOps project has as target to improve automated support technology. One of the points to improve is create a better situational awareness and allow better anticipation on things to come. To allow better anticipation, for on the case of our project operators, they would need more information and so build up a better situational awareness. If too much information is supplied information overload would occur. Information overload is a danger that occurs in the current situation, where there is alot of information available. Currently information (chat or email) directed to a single reciever is send to a very large group of people, so those receivers could anticipate on that information. This means an operator also receives (alot of) information that is not relevant for him. To prevent this risk of information overload, automated determination of relevance of information, and the display of this filtered information is one of the projects under the SmartOps project.

The target is to prove it is possible to create an application that determines the relevance of information for specific actors during a military mission, and to analyze the effect of providing this information to the actor. A more specific definition can be found in Appendix A. To reach this we develop an algorithm to determine the relevance of information provided. We also look at the way of displaying this information. At the end of the project we do a pilot to determine the effect of providing filtered information. We also produce multiple deliverables to show the progress of our project, including a test plan. At the end we shall produce a final report and give a presentation.

The products that shall be produced will be an application that determines the relevance of information, and shows the relevant information to the user. Furthermore we shall produce a conclusion based on the pilot done with the application, and a (design of a) presentation about the results of the pilot and the effects of the application.
3 Approach

This section describes how the project will be executed. The approach should facilitate that the stated goals are met during the execution.

3.1 Method

Methods are the framework that will be used to execute the project, to give a template for the different phases and their connections.

A method which is suited for a research software project is essential. Several questions are raised in the project assignment. Questions raised are:

- What information experts find relevant during a mission?
- What factors play a role in deciding whether information is relevant?
- What are the effects of giving too much information?

These questions are still unanswered. It is still not decided how the information filtering will be accomplished and what the effects will be. Therefore, these questions must be researched using a suitable method which takes this research into account.

TNO has done many research projects and has experience using different methods. TNO has used the Situated Cognitive Engineering (sCE) methodology successfully in the past. For this reason, our TNO mentor has required this methodology.

The sCE methodology was developed by M.A. Neerincx et al. The sCE methodology consists of several phases:

- Derive — An “integrated analysis of the operational, human factors and technological drivers or constraints” [1] is made.
- Specify — From the derivation phase, use cases are made. Furthermore, claims are made “to justify design decisions, highlighting the upsides, downsides and trade-offs involved”[2]. Requirements are formulated, justified on the claims and contextualized on the uses cases.
- Test — In the test phase, a human-in-the-loop test can be set-up with a prototype.
- Refine — On the basis of the test results, refinements are made to all products produced in the previous phases. Therefore, claims can be adjusted because of new insights and the requirement can change accordingly.

The “[...] process is highly iterative with incremental top-down developments of functions.” [1] Looking at the sCE phases, it can be concluded that the focus is on a making justified requirements, which makes it a suitable methodology for research projects. In this project, the sCE methodology will give a way to answer the questions previously stated in a justified way and will ensure that requirements and goals will be met.

3.2 Emphasizes the Proof of Concept

Since this is a research project, no priorities are on acceptation, since the system produced will not be used in the field. However, the projects advantages for actual use should be clear to members of the Ministry of Defense, to guarantee further development. Therefore, a presentation will be held for them. During the design of the prototype this presentation should be kept in mind.

4 Project Organisation and Conditions

The project is partially executed at TNO Soesterberg and partially at TU Delft. We visit TNO Soesterberg two days per week, where we will be treated as guests. The other days we work at the TU Delft, in a project room.
A lot of work has to be done in a short time to finish the product and its reports. In order to keep control of the process we plan to have several meetings each week. In these meetings we will discuss the current process and create a planning for the next week. There will be at least three contact days with our TNO mentors, including a weekly meeting every Tuesday. Every Friday there also is a weekly meeting arranged with our TU Delft mentor. We will produce an agenda and minutes for these official meetings.

Both TNO and TU Delft have different requirement conditions for this project. The TU Delft requires us to deliver several documents. These include a project initiation document (this document), an orientation document, and a document with use cases and requirements, including an architectural and technical design. TNO Soesterberg requires us to follow the sCE approach (see also Section 3.1) during the project. TNO Soesterberg also wants to execute a pilot experiment and prepare presentation documents for a TNO internal presentation. Finally we will present the results of the project at the TU Delft.

During this project we will work on our own available hardware. At TNO Soesterberg we have only one internal network connection.

5 Project Planning

The planning can be found in Appendix B. We have examened the schedule of last years TNO group and combined the result with our own experience to create the planning. The project starts with research being conducted and documented, providing a base to design the application and decide on the exact project scope. When a good foundation is put together for the design and agreed upon the implementation phase will start. The architectural design document and design document will provide us guidance to ensure a smooth implementation phase without big surprises. At the end of the implementation the Software Improvement Group will review our code. During the whole project we will extend and update our final report. After the implementation a pilot is held at TNO and recommendations are added to the final report. The project is finished with a presentation at TNO and at TU Delft by our group.

6 Quality Control

In the project several tools will be utilized to ensure the right quality of the final product. A combination of clear and frequent communication combined with a pilot and thorough testing will provide us with control on the project quality.

6.1 Communication

Multiple communication channels are used during the project to ensure the client, project group and mentor are all on the same page. By maintaining regular communication small adjustments can easily be made to steer the project in the right direction, thus not requiring any major changes and less sudden increases to the workload.

6.2 Pilot

Towards the end of the project a pilot is conducted at TNO to test with users and receive feedback. One of the main goals is to see what the improvement is of our prototype to the relevance of the information received. Although there will be no time to implement the feedback the major goal of the pilot is to indicate where and how future research can improve the application.

6.3 Testing

The final tool that is used is the art of testing. Stability and efficiency are not our main goals as the resulting application will only be a prototype. That’s why testing will be focused on high level tests to ensure the designed algorithm is working correctly but also to determine the boundaries and scope of our algorithm.
A Project Description (Dutch)
1. Inleiding
De beschikbaarheid van (digitale) informatie neemt steeds verder toe in onze wereld en biedt veel nieuwe mogelijkheden. Een probleem is echter dat deze mogelijkheden alleen benut kunnen worden wanneer de informatie op het juiste moment bij de juiste persoon terecht komt, in het juiste format. Op dit moment wordt hier vaak te weinig rekening mee gehouden (figuur 1).

![Figuur 1: Overflow aan informatie, op de verkeerde manier aangeboden.](image)

Ook bij militaire operaties heeft een toename van de automatisering gezorgd voor een toename in digitale informatie. Op dit moment bepaalt de opsteller (mens of systeem) van deze informatie voor wie deze informatie bestemd is, waarna er een bericht aan deze geadresseerde verstuurd wordt. Daarnaast kan soms het bericht nog ‘ter info’ aan anderen verstuurd worden. De informatie is dan niet in de eerste plaats opgesteld voor deze doelgroep, maar mogelijk wel nuttig om te weten. De bepaling voor wie informatie “ter info” relevant kan zijn is lastig, maar uitermate belangrijk. Wanneer mensen te veel berichten ontvangen die niet relevant zijn voor hen, wordt er niet meer naar gekeken; de informatie wordt van te voren al afgedaan als irrelevant. Wanneer er te weinig mensen toegang tot de berichten krijgen gaat de kracht van belangrijke informatie meteen al verloren.

Deze opdracht bestudeert hoe relevantie van informatie bepaald kan worden in een militaire setting. Daarnaast wordt er gekeken naar de effecten van het ‘ter info’ aanbieden van relevante informatie.
2. Opdracht
Met deze opdracht willen we inzicht krijgen in de relevantie van informatie tijdens militaire missies. Dit doen we met behulp van een experiment met experts uit het militaire domein die een realistisch scenario spelen (figuur 2). Mogelijke vragen voor deze opdracht:

A. Welke informatie vinden experts tijdens het verloop van een missie relevant? Welke factoren spelen hierbij een rol?
B. Zijn deze factoren tijdens een missie te bepalen uit berichten en meta-informatie? Is de manier waarop de relevantie van deze factoren afhangt te modelleren, zodat de relevantie voorspeld kan worden?
C. Wat zijn de effecten van het verstrekken van relevante informatie? Hoe wordt informatie met verschillende relevantie aan de gebruiker gepresenteerd? Wordt minder relevante informatie bijvoorbeeld anders gepresenteerd? Of helemaal niet? Waar liggen de grenzen?

Figuur 2: Een expert van de marine voert een scenario uit achter de testopstelling bij TNO.

Tijdens de opdracht moet er gedacht worden aan de volgende werkzaamheden:
A. Een verkenning van het begrip relevantie in literatuur en op internet.
B. Ondersteuning bij het verzamelen van data tijdens een experiment.
C. Het opstellen, trainen en testen van een relevantie-model. Denk hierbij aan begrippen als fuzzy logic, classificatie algoritmes en machine learning.
D. Het testen van het model met domein experts.

Het is mogelijk de opdracht af te stemmen op individuele wensen, mede afhankelijk van het aantal deelnemende studenten. Er kan per direct gestart worden, de opdracht zal in ieder geval eind Juli 2011 beëindigd moeten zijn. Uitvoering in overleg bij TNO in Soesterberg of bij de TU in Delft. De voorkeur gaat uit naar Nederlandstalige studenten (in verband met een Nederlandstalig experiment).

Contact en meer informatie bij:
Marc Grootjen, marc@grootjen.nl, +31647484098
Wouter Vos, Wouter.vos@tno.nl
B Schedule
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Action</th>
<th>Deliverable (end of day)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18-04-11</td>
<td>Kick off</td>
<td></td>
</tr>
<tr>
<td></td>
<td>19-04-11</td>
<td>Read Literature</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20-04-11</td>
<td>Read Literature</td>
<td></td>
</tr>
<tr>
<td></td>
<td>21-04-11</td>
<td>Write orientation</td>
<td>Concept Orientation + PID</td>
</tr>
<tr>
<td></td>
<td>22-04-11</td>
<td><strong>Good Friday</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>23-04-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>24-04-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>25-04-11</td>
<td><strong>Easter Monday</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>26-04-11</td>
<td>Write orientation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>27-04-11</td>
<td>Write orientation</td>
<td>Final Orientation + PID</td>
</tr>
<tr>
<td></td>
<td>28-04-11</td>
<td>Prepare interview</td>
<td></td>
</tr>
<tr>
<td></td>
<td>29-04-11</td>
<td>Interview with user + D&amp;S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30-04-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01-05-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>02-05-11</td>
<td>D&amp;S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03-05-11</td>
<td>D&amp;S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>04-05-11</td>
<td>D&amp;S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>05-05-11</td>
<td><strong>Freedom Day</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>06-05-11</td>
<td>D&amp;S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>07-05-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>08-05-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>09-05-11</td>
<td>D&amp;S</td>
<td>D&amp;S Documents</td>
</tr>
<tr>
<td></td>
<td>10-05-11</td>
<td>ADD</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td>11-05-11</td>
<td>TDD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12-05-11</td>
<td>TDD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13-05-11</td>
<td>TDD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14-05-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15-05-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>16-05-11</td>
<td>TDD</td>
<td>TDD</td>
</tr>
<tr>
<td></td>
<td>17-05-11</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18-05-11</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>19-05-11</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20-05-11</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>21-05-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>22-05-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>23-05-11</td>
<td>Implementation</td>
<td>Hand in code to SIG</td>
</tr>
<tr>
<td></td>
<td>24-05-11</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>25-05-11</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>26-05-11</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>27-05-11</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>28-05-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>29-05-11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>30-05-11</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>31-05-11</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>01-06-11</td>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Event Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02-06-11</td>
<td>Ascension Day</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03-06-11</td>
<td>Ascension Day</td>
<td></td>
<td></td>
</tr>
<tr>
<td>04-06-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05-06-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06-06-11</td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>07-06-11</td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>08-06-11</td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>09-06-11</td>
<td>Implementation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-06-11</td>
<td>Bugfix and test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11-06-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-06-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13-06-11</td>
<td>Second Pentecost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14-06-11</td>
<td>Bugfix and test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15-06-11</td>
<td>Bugfix and test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-06-11</td>
<td>Prepare pilot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17-06-11</td>
<td>Prepare pilot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18-06-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19-06-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20-06-11</td>
<td>Pilot</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21-06-11</td>
<td>Write recommendations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22-06-11</td>
<td>Prepare TNO presentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23-06-11</td>
<td>Finish end report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24-06-11</td>
<td>Finish end report</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25-06-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26-06-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27-06-11</td>
<td>Prepare presentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28-06-11</td>
<td>Prepare presentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29-06-11</td>
<td>Prepare presentation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30-06-11</td>
<td>Finish</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01-07-11</td>
<td>Finish</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D&S = Derive & Specify  
PID = Project Initiation Document  
SIG = Software improvement Group  
ADD = Architectural Design Document  
TDD = Technical Design Document
References

Orientation Document

Bram Beernink  
Arjen Goedegebure  
Niels van Kaam  
Remco van der Zon  

Summary

This document contains a report of the basic orientation we have done at the beginning of our Project. At the first chapter we describe the client, TNO, for who we will execute the project. Then we summarize the contacts we have for the project at both TNO and TU Delft. Next is a short description about the problem and the goals we want to reach. In chapter 5 we give a list of the deliverables we produce during the project. Thereafter we summarize the preconditions under which we execute the project, and last we show the implications if we fail to complete this project.

1 Client

TNO is an independent research organization that focuses on applying scientific knowledge to practical situations. Founded in 1932, TNO has a rich history of solving problems with research. With about 5000 people currently employed TNO is the largest research organization in the Netherlands. TNO has focused all its knowledge into three main divisions:

- Technical Sciences
- Behavioral and Societal Sciences
- Earth, Environment and Life Sciences

The mission of TNO is to connect people and knowledge to create innovation that enhances the competitive power of companies and durable increase the health of our society.

2 Contacts

In the project we will work directly with two representatives of the client TNO. Besides TNO also the TU Delft will keep regular contact and provides us with supervision during the project.

2.1 TNO

Ir. M. Grootjen (marc@grootjen.nl) — Supervisor  
Ir. W.K. Vos (wouter.vos@tno.nl) — Supervisor

2.2 TU Delft

Prof. dr. M.A. Neerinck (mark.neerinck@tno.nl) — The direct mentor from TU Delft  
Drs. P.R. van Nieuwenhuizen (P.R.vanNieuwenhuizen@tudelft.nl) — Coordinator of bachelor projects
3 Problem Description & Goals

3.1 Problem Description

Because of the automation nowadays much more information is available to people. Also during military missions more information is produced. At this moment the creator of the information determines the receivers of the information. The determination of who will receive the information is important. If an operator receives too much information, of which many is not relevant for him, information overload might occur. If an operator receives too few information, he might miss information that was of good use. Therefore might be usefull to determine the relevance of information, so an operator receives filtered information.

Therefore our mission is to show it is usefull to investigate further into the automated determination of relevance of information during military missions.

3.2 Goals

We have split up the mission into two main goals. The first goal is to produce an automated way to determine the relevance of information during a military mission for a specific operator. If we are able to do so, we would prove it is possible to automatically determine relevance. The second goal is to design a userinterface, focussed on keeping the workload as low as possible. As a third subgoal we analyse the effect of providing the produced relevant information to the actor. Because of the short time we probably will not be able to fully complete this third goal.

4 Products Produced

In this section, the products produced during the project will be listed. The end result will be a working software prototype which will be able to filter relevant information for the Helicopter Direction Officer (HDO).

For a large part, the sCE methodology dictates what kind of products should be produced to come to the end result. For each phase of the sCE methodology, the following products will be produced:

<table>
<thead>
<tr>
<th>sCE phase</th>
<th>Products produced</th>
<th>Sections</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Derive</td>
<td>Analyses of the Current Situation</td>
<td>• Operational Demands</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Human Factors Knowledge</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Envisioned Technology</td>
<td></td>
</tr>
<tr>
<td>Specify</td>
<td>Specification</td>
<td>• Use cases</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Claims</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Requirements</td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td>Prototype</td>
<td>• Test Plan document</td>
<td></td>
</tr>
<tr>
<td>Refine</td>
<td>Prototype Review</td>
<td>• Pilot review</td>
<td></td>
</tr>
</tbody>
</table>

The sCE methodology does not focus on the software engineering process itself. Therefore, several additional documents will be produced, which focus on the software development process. These additional documents produced will be:

- User Interface Design Document —
- Architectural Design Document (ADD) — This document will focus on the high-level architectural software design.
- Technical Design Document (TDD) — This document will focus on
- Test Plan
5 Literature research

One important requirement by both TNO and the TU Delft is a literature research done on the subject. Most, but not all, literature research is done during the (sCE) derive phase. This phase is reported in the final report, in the chapter “Analysis”.

6 Preconditions

In order to complete our prototype product, we have defined some preconditions. These defined preconditions make sure we can focus on the message filtering and message representation. Some of these preconditions are only valid because we are going create a prototype which should test the method of filtering.

- All communication is text-based, and is executed over this system only.
- There is a network connection available between all actors. This network connection is secure, and no network failures occur.
- Metadata is already appended to a message. This includes the sender, locations, and more.
- The actors metadata is available to the system, including the actors location, rank, and more.
- This prototype does not need to be integrated with currently used systems.

7 Implications of Failure

There are some consequences for failing the project that will be bundled by group. As stated before the goal of this project, for TNO, is to show the potential of the idea of relevance and to have this project nominated for more funding and further research. TNO will need to have the right material to convince that there is true potential in applying relevance. Any failure in this area will reduce the chance of more research to relevance.

One of the major requirements of the TU Delft is to have a working prototype at the end of the project. Besides the prototype the TU Delft also focuses on the process of how the prototype was created. In case of failing on the prototype or the process followed during the project it will not be possible to get a sufficient grade.
Introduction

The architectural design document (ADD) is used to identify all the required system components and their uses. The system components are deducted from the requirements, use cases and claims. The information gathered in the research phase is used to fill in the system components and link the components. By creating the ADD we ensure that all components that are needed are developed.

1 System components

In this section the main system components will be determined. We will do this by exploring the requirements that were set up in the derive and specify document. In Section 5 it is described how these components are related to each other and how they communicate with each other.

First the different system components as drawn in Figure 1 will be explained. The operators are going to use an App with an user interface (requirements 2, 3, 6, 7, 8, and 12). From the UI we derive the first system component: the UserApp.

Since the system has to handle messages, the messages should be easily retrievable (requirements 6, 7, and 10). For this purpose we use a data storage component. For a more in depth discussion on which data storage is going to be used, see Section 3. For now it suffices to state that Sesame was used for data storage.

For testing purposes, the system also needs a component which puts all send messages in the database, at the time the message was sent (requirement 9). This will be the message parser. For the system that will be made by us, this will be the only way messages are entered in the system. Users cannot enter messages themselves.

The relevance for a message per person has to be determined, we will call this the Relevance Algorithm. The Relevance Algorithm consists of a fuzzy expert system and the LSI algorithm (requirements 11, and 13).

Finally, the ticker is the module that triggers the relevance algorithm to re-rate a message if it gets older.

Besides the core system a number of modules is needed to create a fully functioning system. The mission task manager is the module where the mission task tree is sent to the database and changes can be made to the mission task tree. The role manager will manage the names and creation of military roles, such as the HDO and SENS. The position emulator will emulate the position of these roles. For example, it will continuously update the longitude, latitude and position of the HDO. The mission task manager, role manager and the position emulator will not be implemented as with the limited amount of available time we will focus on the Relevance Algorithm, data storage and UserApp first.

We have now defined the following system components:

- Relevance Algorithm
- Sesame Data Storage
2 Relevance Algorithm

The relevance algorithm is the component that determines the relevance of a message for the operators, and stores the value in the database. The algorithm contains two sub-components, Latent Semantic Index (LSI) and the fuzzy expert system. The LSI component determines which messages belong to each other, based on the content of the messages. By looking at the relevance of messages that are close to the new message, through using LSI a certain rate of relevance can be determined. This result is then used as input, together with other metadata, to the fuzzy expert system to determine the relevance of a message for a specific operator. A description of each of these components follows.
2.1 LSI

As was found in the envisioned technology section of the soC methodology, Latent Semantic Indexing (LSI) will be used to determine which messages belong to each other. This provides a way to make sense of the content of messages. By looking at the relevance of messages close to the given message a new relevance value can be determined. If the user provides a relevance value for messages, these messages can be taken up into LSI and the feedback is used to determine the relevance of new messages.

Several frameworks exist, that are able to index text using LSI. Apple, Inc. has made the Latent Semantic Mapping\footnote{http://developer.apple.com/library/mac/#documentation/TextFonts/Reference/LatentSemanticMapping/} framework available. A disadvantage is that it is closed source, the current development of the framework is low and the framework is not very flexible. Tweaking of the algorithm is not possible.

Gensim\footnote{http://code.google.com/p/airhead-research/} from Radim Rehůřek is a Python framework for Vector Space modelling. This could also be used for LSI, but it requires manual parsing of text messaging and representing the messages in vector space, before Gensim can be used. This requires significant extra work, if text parsing is to be done complete with preprocessing.

S-Space Package\footnote{http://mlp.muni.cz/projekty/gensim/} by airhead-research is Java software library which can be used for LSI. It is open source, is stable but also under active development. Additionally, it supports different algorithms, so it is possible to tweak the framework to the needs imposed by the environment.

The clear advantages of S-Space Package is the reason for choosing it as a software library for LSI.

2.2 Fuzzy Expert System

The software will have a fuzzy expert system implementation. The fuzzy expert system has a set of rules and produces a relevance value for each operator. It is possible to filter and sort messages based on the determined relevance, since the relevance value is a value between 0 and 1. Fuzzy expert systems has been a subject for research for several years, this results in the availability of many frameworks. In order to choose the best framework, the frameworks are compared on available platforms, rules storage, possible integration with other software, activity, documentation, and license. We have listed jFuzzyLogic\footnote{http://jfuzzylogic.sourceforge.net/html/index.html}, Jess\footnote{http://www.jessrules.com/}, FuzzyCLIPS\footnote{http://awesom.eu/~cygal/archives/2010/04/22/fuzzyclips_downloads/index.html(mirror)}, D3Web\footnote{http://sourceforge.net/apps/mediawiki/d3web/index.php?title=Main_Page}, Fuzzy Prolog\footnote{http://www.ciaohome.org/ciao_html/ciao_113.html#SEC467}, and AI:fuzzyinference\footnote{http://search.cpan.org/~aqumsieh/AI-FuzzyInference-0.05/}. The results can be seen in Table 1.

jFuzzyLogic, Jess and AI:fuzzyinference seem fair choices because they are up to date and support fuzzy data. jFuzzyLogic and Jess are preferred over AI:fuzzyinference because they can be easily integrated in Java. Jess has better documentation than jFuzzyLogic, but we have to place a request to download Jess. Because time is of the essence we decided to chose for jFuzzyLogic.

<table>
<thead>
<tr>
<th>Language/Platform</th>
<th>jFuzzyLogic</th>
<th>Jess</th>
<th>FuzzyCLIPS</th>
<th>D3Web</th>
<th>Fuzzy Prolog</th>
<th>AI:fuzzyinference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports fuzzy data</td>
<td>Yes</td>
<td>Yes, by a plug-in</td>
<td>Yes</td>
<td>?</td>
<td>By the Ciao Prolog module</td>
<td>Yes</td>
</tr>
<tr>
<td>Rules in code or external file</td>
<td>Both</td>
<td>External</td>
<td>External</td>
<td>External Java</td>
<td>Code</td>
<td>Code</td>
</tr>
<tr>
<td>Integration with software</td>
<td>Java A.P.I.</td>
<td>Java A.P.I. or own</td>
<td>Own</td>
<td>Own</td>
<td>Own</td>
<td>Own</td>
</tr>
<tr>
<td>Documentation</td>
<td>Poor</td>
<td>Yes</td>
<td>Yes</td>
<td>A lot</td>
<td>A lot</td>
<td>Yes</td>
</tr>
<tr>
<td>License</td>
<td>LGPL</td>
<td>Non Commercial free (on request)</td>
<td>Non Commercial free</td>
<td>LGPL</td>
<td>LGPL</td>
<td>?</td>
</tr>
</tbody>
</table>

Table 1: Overview of fuzzy expert systems and their main properties.
3 Data storage

The system will receive messages from different users. These messages need to be stored and retrieved somehow, as described in Section 1. A common practice to do this is using a data management system. There are a lot of data management systems available.

Each data management system has certain properties and restrictions. In order to decide which database is best to use, the main properties of most popular databases are listed in Table 2. We have chosen to have a look at SQLite, MySQL, PostgreSQL, CoreData, and Sesame.

In the first column the properties are listed. First the query language and available platforms for the database system are listed. Next ORM support is evaluated. ORM stands for “Object-related mapping”, and in general it enables the developer to access database-rows as objects, and vice versa, for object-like database usage. Next it is stated if the database supports RDF (resource description framework). RDF is defined as follows: “RDF extends the linking structure of the Web to use URIs (Uniform Resource Identifier) to name the relationship between things as well as the two ends of the link (this is usually referred to as a “triple”). Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications.”[1]. We check the support for spatial data queries; it may be useful to get posted messages of a specific area on the real world map. Continuous queries are queries which reports periodically changes in the database-data done by other application.

<table>
<thead>
<tr>
<th></th>
<th>SQLite</th>
<th>MySQL</th>
<th>PostgreSQL</th>
<th>CoreData</th>
<th>Sesame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query-language</td>
<td>SQL</td>
<td>SQL</td>
<td>SQL</td>
<td>By predicates</td>
<td>SPARQL</td>
</tr>
<tr>
<td>Platforms</td>
<td>All</td>
<td>All</td>
<td>All</td>
<td>Mac OS X / iOS</td>
<td>All (Java Apache/Tomcat)</td>
</tr>
<tr>
<td>ORM Support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RDF Support</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Spatial data Support</td>
<td>No</td>
<td>Yes, with spatial extension enabled plugin</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Continuous queries</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>No</td>
</tr>
<tr>
<td>System</td>
<td>Local</td>
<td>Server</td>
<td>Server</td>
<td>Local</td>
<td>Server</td>
</tr>
<tr>
<td>Installation</td>
<td>Application</td>
<td>Application</td>
<td>Apache/Tomcat</td>
<td>Plugin</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 2: Overview of database properties.

From the database properties presented in Table 2 one can conclude that SQLite, MySQL and PostgreSQL are very similar to each other. Sesame differs from the databases just mentioned because the data is stored in an RDF format. At TNO Soesterberg an research program, Mutual Empowerment, is using Sesame RDF. In the way Sesame is used, rapid prototyping is possible. Being compatible with the Mutual Empowerment system enables future research programs to use the results of this project in an easy way. TNO has stated that it would be an advantage if Sesame was used. Because we could see the advantages for TNO, and Sesame support was beneficial to the continuation of further research, Sesame was chosen for data storage for our project.

4 Sesame libraries

In Section 3 we concluded that we will use Sesame as the database management system. In this section we will explore which library can be used to communicate with the database. The community using the Sesame library has created a wide variety of libraries to support different programming languages. We have chosen to have a look at OpenRDF, DotNetRDF, PhpSesame, and RdfLib.

TNO’s Mutual Empowerment group has developed a Sesame library named TrexCore. It allows one to easily set up a prototype [3]. This system allows different prototypes to share their data and

---

10http://www.sqlite.org/
11http://www.mysql.com/
12http://www.postgresql.org/
14http://www.opendlg.org/
15http://www.openrdf.org/doc/sesame/users/ch01.html/
16http://www.dotnetrdf.org/
17https://github.com/alesslatchford/phpSesame/wiki/
18http://www.rdflib.net/
certain views at a database level instead of writing an *Application programming interface*. These views are graphical representations of a data-set. Because TNO has requested to consider using Trexoore we also include this system to our comparison.

The table below gives an overview of the mentioned libraries and their features.

<table>
<thead>
<tr>
<th>Library</th>
<th>Language</th>
<th>Continuous Queries</th>
<th>Maturity</th>
<th>Last Update</th>
<th>Support</th>
<th>Active Community</th>
<th>Open Source</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenRDF</td>
<td>Java</td>
<td>No</td>
<td>Release</td>
<td>07/04/2011</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>DotNetRDF</td>
<td>C#</td>
<td>No</td>
<td>Beta</td>
<td>01/03/2011</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PhpSesame</td>
<td>PHP</td>
<td>No</td>
<td>Alpha</td>
<td>21/03/2011</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>RdfLib</td>
<td>Python</td>
<td>No</td>
<td>Release</td>
<td>???</td>
<td>No</td>
<td>Yes, personal</td>
<td>Yes, open</td>
<td>Yes, extensive</td>
</tr>
<tr>
<td>Trexoore</td>
<td>C#</td>
<td>Yes</td>
<td>Beta</td>
<td>???</td>
<td>Yes</td>
<td>No, but good cont-</td>
<td>No, but open</td>
<td>Yes, but no extensive</td>
</tr>
</tbody>
</table>

Table 3: Overview of library properties.

Sesame itself has a Java library with extensive documentation. OpenRDF is continuously updated and has a high maturity level in comparison to the C# and PHP libraries.

The C# library dotNetRDF is a powerful tool using the latest .Net version with an active community. However the library is still in the beta version and bugs are found regularly.

PhpSesame is a very new library that has only just entered alpha phase. The main reason we investigated the PHP library is to see if PHP would be a viable program language. Although the developments are interesting to follow phpSesame is not a contending candidate.

RdfLib for Python was also considered. The RdfLib is just as the openRDF library for Java already in the release state and has a large community.

Finally, Trexoore was considered. Trexoore has several advantages. Since Trexoore is a TNO project, support is good, because contact with the developer is good. Feature request can also be made. Furthermore, Trexoore supports continuous queries. As will be discussed in Section 5, this is a clear advantage.

However, Trexoore was written in C#. If we combine our conclusions from Section 2.1 (LSI) and Section 2.2 (fuzzy expert systems), a Java library is preferred. It saves large amounts of time if the LSI and Expert System are written in the same language as the Sesame library. We contacted the author of Trexoore to see the possibilities in regard to a Java implementation but unfortunately we would have to port Trexoore ourselves. Additionally, Trexoore has no ORM support, while this would be highly beneficial.

Therefore, it was decided to write our own library for Sesame communication, with support for continuous queries and simple ORM support. The developed library will use Sesame’s Java library for direct communication with Sesame. In this way we use Sesame, but still use the flexibility of the Trexoore system as TNO has requested.

5 Stand-alone software application

As can be seen in the system architecture (Figure 1), data storage is central in a star lay-out. All data sharing between the different components will therefore go via this central data storage. This allows the different system components to be implemented as stand-alone software applications. The only link between the different software applications will be the data storage. Still, different software components need different data when something changes in the central data storage. In other words: the different software components need to be informed when data changes happen which are relevant to that component.

To give a general idea of the flow in the system, several Unified Modeling Language (UML)\(^\text{19}\) communication diagrams have been made. For each database update of each system component, a diagram was made.

\(^{19}\text{http://www.uml.org/}\)
In Figure 2, a message is first put in the database by the message parser. The relevance algorithm picks up the unrated message and assigns a relevance to the message. Next, the user interface gets the rated messages to display to the user.

In Figure 3, the age of a message is updated by the ticker. The relevance algorithm picks up this update and assigns a new relevance to the message. Next, the user interface adjusts the rating of this message and displays this to the user.

In Figure 4, a mission task is updated by the mission task manager. The relevance algorithm is notified of this update and adjusts message relevances if necessary. Next, the user interface adjusts the rating of these messages and displays this to the user.

In Figure 5, a user role is updated. Since this does not affect relevances, the Relevance Algorithm is not notified. However, the UserApp will process this change, to display the new role name. Also the mission task manager is notified, so it will display the new role name.

In Figure 6, a position of a role is changed in the database by the Position Emulator. The relevance algorithm is notified of this update and adjusts message relevances if necessary. Next, the user interface adjusts the rating of these messages and displays this to the user.

Several techniques are available to enable the different components to be notified when data in the database has changed. One option is to let each component periodically poll the data storage and compare the data, with the previous situation. Another option is to let the database inform the components of changes.

Sesame does not provide this last option. As was previously stated, it was therefore chosen to write our own library in Java. This library allows all components to query the database. The library will then only give back the added messages and messages which attributes have been changed. This will be accomplished using continuous queries. This makes the individual components more modular.

For example, the relevance algorithm can continuously query for all messages in the database. The library will then only give back the added messages with a notification, which the relevance algorithm will then be able to process and give a relevance attribute for this added message for each operator.

Figure 2: Component communication when a new message is inserted.
Figure 3: Component communication when the a message age changes.

Figure 4: Component communication when the missions tasks are updated.
Figure 5: Component communication when a role is updated.

Figure 6: Component communication when a role position changes.
References


[3] M. van Zomeren, “Mutual empowerment prototype development kit (me-pdk).” This is an unversioned draft user manual for TrexCore. In order to retrieve this document, one may contact the authors at TNO Soesterberg.
Introduction

The technical design document (TDD) contains all details regarding the design of the different software components. The TDD refines the design of the system components from the Architectural Design Document (ADD). For the database component, the TDD contains the database ontology. For all other system components which will be implemented and the database library, class diagrams are made. Continuing on the class diagrams, the used design patterns are stated. The last paragraph contains the specifications of all classes.

1 Database Ontology

Following the conclusion on the database in the ADD we use Sesame, which is a RDF type database. This means that the database uses an ontology that dictates the structure. In this section the ontology followed by our application is given. To give a better understanding, the ontology has been cut in multiple pieces to highlight the main parts of the database. The different pieces are the role, message, and the relevance part. The images in this section are only useable for human readability. In Appendix A a more detailed image is given, which can be used by developers to see where the data is stored in the database.

First, the role part of the ontology (Figure 1) is where the role of a person is defined with all its attributes. A role is defined by its name, movement, position, and distance to other roles. A role has no person attached to it in the database. As the algorithm will not make any distinction between different persons fulfilling the same role. The choice was made to leave out a person object in the database, to save time and make the queries more straightforward. The movement and position of a role are the movement and position of the person that is fulfilling the role at that moment. The movement is defined by the speed and direction. The position is defined by longitude, latitude, and altitude. Finally to assist the algorithm in determining the relevance of a message the distance between roles is defined. As an example HDO (Helicopter Directie Officier) has a low distance to helicopters that are flying but might have a high distance to land units.
Second is the message and its attributes (Figure 2). The message has a position. This position is defined as the position of the role that created the message when it was send. Classification is used to determine if the message is about an friend, neutral or hostile entity. The priority of a message and the number of times read help the algorithm determine the importance of a message. The time stamp is set when the message is send and the age field will be set to fresh. Overtime the age changes to young and old. The subject and content contain the actual information that is transmitted by the author of the message. Finally the message has a link to the role that is the author of the message and a link to the role(s) that received it.

The relevance is a value that links a message to a person (Figure 3). There are three different relevance
values. The relevance value is the value calculated by the algorithm. The relevance value by user, is input from the user that is used to measure how much the standard relevance value matches the users idea of relevance. Finally there is the relevance value were the time doesn’t have any influence. This relevance value will be used if a User Interface (U.I.) is created were the time relevance is embedded in the U.I.. A role will have a relevance value linked to every message and a message will have a relevance value linked to every role.

Figure 3: Ontology of Relevance.

The mission tasks will not be part of the database as of the current version. The mission task tree is not ready to be used and it saves time to use a table within our code.

2 Class Diagram

In this section all class diagrams are listed. The class diagrams are noted in standard UML notation and were written using the astah application.

As can be seen in the Architectural Design Document (ADD), three separate applications will be implemented. The following applications will be implemented:

- User Application
- Message Generator
- Relevance Algorithm

In the ADD it was specified that all these applications will have a database connection to a Sesame database. This connection will be managed by our own library, Trex Core Relevance (TCR). TCR is primary used to connect the database and map simple objects to database tuples. These objects can be retrieved through a query given to a session. Pushing, updating and removing these simple objects can be done by using a transaction. Once a transaction is committed a set of change operations is performed on the database. The TCR class diagram can be found in Figure 4, the simple data objects class diagram can be found in Figure 6.

In order to hold relations between objects another package using TCR is built. This package, Overlaid Trex Core (OTC), handles relations between objects and makes sure only needed relations will be loaded. OTC also contains data classes which will be used in the other application components. A class diagram of the OTC package can be found in Figure 5.

Note the OTC classes extending OTCModel have relations to the TCR data classes, as defined in Figure 6. For example, an OTCMessage is composed of a TCR Message. Many getters and setters of OTCMessage will directly edit the TCR Message, such as for the message content. However, some of OTCMessage’s methods will not directly edit just its TCR Message, but also other TCR Models.
example, when the relevance of an OTCMessage is set for a OTCRole, this will not only change the TCR Message, but will also create or change a TCR MessageRelevance.

As can be seen the reason to create the OTCModel and TCR models is to make sure you don’t have to worry about managing relations when using functions from OTCModel. This enabled us to set up a very easy Ticker as OTC took care of the bulk of the work of connecting to the database.

Also note that the user application class diagram (Figure 7) and the relevance algorithm class diagram (Figure 9) have relations to the classes extending OTCModel. These relations are not listed in the class diagrams since the connection between each application and the data classes is straightforward. Listing each of these classes per application would not be meaningful. Therefore, these data classes are listed separately.

2.1 Used Design Patterns

2.1.1 Command Pattern

In the Command design pattern [1], a separate object, called the invoker, is used to encapsulate all changes made to a receiver object. This pattern has several advantages. The primary reason for using this pattern is because transactions can be implemented using this pattern. Transactions are atomically executed modifications on data. The invoker can bundle multiple commands and when a commit is done, the transaction is executed. Moreover, the invoker can simply keep track of all changes made, so undo functionality can be added.

In the design of TCR and OTC the influence of the Command design pattern is visible. A Transaction can be viewed as the invoker. It keeps track of changes, so transactions are implemented. It would be easy to add undo functionality to this class.

2.1.2 Observer Pattern

In the Observer design pattern [2], an Observer observers an Observable. Observables extend a generic Observable class, which provides the infrastructure for notification of Observers when an Observable changes. For example, an application’s controller can implements the Observer interface so it can observe the QueryDelegate, which extend the generic Observable class. This way, the QueryDelegate can notify its Observers of changes, such as the User Application which observes the QueryDelegate. This is how all applications will be notified of database updates.
Figure 4: Class diagram the TCR package.
Figure 5: Class diagram of the OTC package.
Figure 6: Class diagram of relations between data classes.
Figure 7: Class diagram of the relevance algorithm.
Figure 8: Class diagram of the user application.
Figure 9: Class diagram of the ticker application.
3 Class Specifications

A Database Ontology – Full names

In (Figure 10) the ontology is shown with the exact links. In the figure can be seen how the RDF triplets are defined and which naming conventions are used. The used naming convention for classes is: http://relevance.com/*class*/id*. All predicates are defined by: http://relevance.com/*class*/#*value*. A value is a field like author for the message class or the longitude field in the function class.
References
