NTNU – Trondheim
Norwegian University of
Science and Technology

Chat Functionality for Communication between Centralized Control Room and Field Operators in a Nuclear Power Plant

Hilde Marie Schade

Master of Science in Cybernetics and Robotics
Submission date: December 2013
Supervisor: Amund Skavhaug, ITK
Co-supervisor: Svein Nilsen, Institutt For Energiteknikk(IFE), Halden

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU – Trondheim
Norwegian University of
Science and Technology

Chat Functionality for Communication between Centralized Control Room and Field Operators in a Nuclear Power Plant

Hilde Marie Schade

December 8, 2013

Assignment

The context of this assignment is the research activity executed to define general recommendations when designing portable units with procedures for nuclear power. This activity are carried out as a part of the OECD Halden Reactor Project, by Institute for Energy Technology (IFE) in Halden. They have for a long time had research activities for computer based procedures for use in centralized control room. This means the defining of guidelines for construction and deployment of computer systems that automate parts of the processes, such as surveillance of process conditions during the execution of certain parts of the processes development. Up until now these research activities have been concentrating on desktop based systems, where these systems is integrated into other systems in the control room.

IFE now have defined a research activity that focus on the deployment of these activities implemented on portable units, per example smart phones, used by the field operators that is doing manual labour in the plant and that cooperate with the operators in the control room. The portable units will be integrated with a desktop system that will be used in the control room. Execution of the processes requires collaboration between the field operators in the control room, which traditionally is done with the use of landlines. The availability of a portable unit for use for the field operator will probably change this collaboration, because one can share both dynamic and static information in a new way. The field operators will have an ongoing status of which parts of the procedures that have been done by the control room and the other way around.

Despite of this it will still be a requirement for immediate dialogue between the control room and the field operator. This can be done with landlines as before, but it is also being considered use of chat functionality. Such a feature would have both advantages as well as disadvantages. One of the supposed advantages is that one is available to get traceable dialogues that everybody can see, and one will easily access other data in the same device. The assumed drawbacks include slowness in the dialogue, and that it will not be as easy to reply with short clarifying questions.

IFEs activities are oriented toward experimental studies in their human-machine laboratory, where they use operators from Nuclear Power Plants that study how

they carry out their assignment, and how it is affected by the availability of the different operator support systems, as computer based procedure systems. Portable units are planned to be tested during 2014. There is a current research activity of designing and developing these portable units. Some are finished, but others remains. One of the remaining is chat functionality. This assignment contains the following elements;

1. A general literature study about human-factors problems related to chat. This study should be completed to interpret established theories into the current context that is the nuclear power industry.
2. A recommendation of a design of the chat functionality for the prototype that is under development.
3. Implementation of chat functionality on an Android unit (which also runs the other parts of the system)
4. A list of hypotheses that can be explored during a future study in HAMMLAB

Preface

This report is written as a Master's Thesis of a Master of Science degree in Engineering Cybernetics, and is an answer to the assignment shown on the page after the title page. This assignment has been written for and carried out as a part of the OECD Halden Reactor Project, conducted by IFE Halden.

First of all I would like to give a big thanks to my supervisor at NTNU, Amund Skavhaug, who despite the physical distance have been a great help with constructive feedback and keeping high spirits. I would also like to thank IFE Halden, represented by Svein Nilsen and the Systems and User Interface Department for housing me during this time and helping with the theoretical and practical parts of the Master's Thesis.

My lovely family deserve a big pat on the back for being there for me during the time of this Master's Thesis. Even though not everybody is still around I would like to state my gratefulness that they all always believe in me, and if needed is never short of encouragement. My enjoyable, funny and weird friends does also deserve a round for being there, not only this last five months, but also during the five years leading up to it.

Abstract

The Nuclear Power Industry nowadays is still operating with communication equipment that does not utilize the technology that has become common the last 20 years. There is a desire to upgrade these communication systems, and a research activity has been started on this task at the OECD Halden Reactor Project. As the communication between field operators and control room still consist of a broadcast/landlines-system a possibility is to add a portable unit with a synchronous messaging feature, commonly known as a chat.

As the consequences of faults in a Nuclear Power Plant is potentially fatal, one of the main considerations, in addition to improving efficiency, is to maintain security. To do so the communication between the control room and field operators need to contain, ideally, no mistakes or confusion. The efficiency is believed to be improved by making it possible for the operators to stay put, answering certain questions from the control room in writing.

To achieve a low rate of misunderstandings and confusion some extra features is added to the application, tested by writers of other articles about chat confusion.

When interviewing operators abut the application there is a bit of scepticism about the need for such an extension. An optimism about communication with multimedia is showed and this is a feature that the operators them selves believe to be useful.

The system made in this article is made as an input to the portable device application to be tested during 2014, and need extensive testing. In order of achieving the goal of no misunderstandings or confusion the application needs to be tested mainly on the amount of use, amount of misunderstandings and how the operators perceive its user friendliness compared to traditional systems.

Contents

Acronyms and abbreviations

IFE	Institute for Energy Technology
HMI	Human-Machine Interaction
UML	User Markup Language
IAEA	International Atomic Energy Agency
NPP	Nuclear Power Plant
FO	Field Operator
CRO	Control Room Operator
VOIP	Voice Over Internet Protocol

List of Figures

List of Tables

Chapter 1

Introduction

When operating a power plant one of the main factors to maintain the high rate of security needed is the communication between the supervisors situated in the control room and field operators inside the power plant. With the severe consequences in mind, if a failure should appear, the nuclear industry is one of the industries that are most conservative and least inclined to adopt new technology. Therefore a potential new product need elaborate testing in research facilities before applied into the every-day routines in a power plant. IFE in Halden is such a research facility where new technology is realised and tested in a virtual control room and with the supervision of industry psychologists.

Even today the only communication between control room supervisors and field operators in a nuclear power plant has been through several installed wired connections inside the plant. When communication was needed, a public intercom system would tell the field operators to pick up the phone. IFE has previously had several studies which defined computer based procedures for use in centralized control rooms. This means the definition of guidelines for construction and deployment of computer systems that is automating selected parts of the procedures. An example of this is surveillance of process conditions during the execution of some parts of the procedures. So far these computer systems have mainly been based on desktop systems embedded with other systems in the control room.

A new study, which is going to be conducted during 2014, will explore the option of including some of the computer based procedures in a portable unit. The current prevalence of portable units has enabled the field operators to communicate with supervisors situated in the control room located outside the plant, from inside the factory without the traditional use of cell phones. These systems will be integrated into desktop systems in the control room.

Availability of such a new hand-held device could have the possibility of altering the cooperation between field operator and control room, caused by the new way

to share information both statically and dynamically. In this way the control room could immediately get notification on what the field operators has done and not done, and vice versa. When these new applications are launched, there is a possibility to renew not only the way to conduct procedures, but also the way the control room and field operators exert the communication among them.

Although spoken communication is the communication the majority of the population have the most training in, there is interest in exploring the possibility to add an online communication system to the program in the portable unit. As most people these days have a substantial amount of training, writing SMSs, this should be a method of communicating which is easy to comprehend. Even so is still a higher probability of misunderstandings than when talking. This report is about the definition of general suggestions of adding this feature, and which measures that needs to be taken to obtain the most natural way of performing written communicating.

This report consists of seven chapters. In chapter two there is basic theory about the human machine discipline, models of how humans make decisions, explanation of how the communication in a nuclear power plant is happening and containing, as well as other chat applications implemented for other purposes. Some of the content in chapter two is the same as in the project thesis;"Improving Remote Control of a Supervising Robot", written by the same author as of this Master's Thesis. Chapter three contains a system suggestion, which founds the base for the implementation of the application explained in chapter four. Some of the results from the theory chapter are discussed in chapter three for the sake of readability. When the implementation was finished there was arranged interviews with a former control operator to determine the most frequently used phrases in the control room. This is explained in chapter five, and is used as a part of the application. Chapter six is discussion around the decisions and results from chapter four, as well as a brief discussion of the results from chapter five. It also contain some discussion related to which hypothesis that should be tested. Chapter six comprises the conclusion of the project, and a suggestion of future work. Parts of the picture on the front page is from [14].

Chapter 2

Theory

2.1 Principle of Human Machine Interaction

When starting to implement a system, one should start by evaluating what the system should do, and make a model of its desired behaviour[15]. One can for example use User Markup Language (UML)-diagrams to decide how to make the best system possible. One of these diagram is a use-case diagram. In this diagram one need to consider how the users will interact with the system, and how they can achieve the best results with it. Sometimes this diagram will reveal that functions that seems as a good idea for the programmer is not necessarily the most beneficial for the users.

The model made before starting to implement the system should resemble the reality as much as possible. Some of the aspects the model should contain is information about the system limits and surroundings, as well as information about whether or not the system is open or closed, the dimensions of the system(purpose, behaviour and structure), analytical models and field studies.

According to Bråthen [16] these issues could be split into two parts: a analytical part, and an experimental part, seen in figure 2.1. The analytical part consist of the "what" and "how" of the problem. In the analysis' what-part the establishment of the demands to the system takes place, as well as a function analysis, and an analysis of the tasks of the operator should be involved in. In this part it is important to give thought to what kind of system that is desired. All demands to the system should be clarified here and several different approaches is possible to evaluate this. One could define different scenarios, map the different things that could happen, make possible scenarios and then define what is needed in these. Another version is to define different users and from there describe what the different users need. Per example will a field operator have different needs in a system than a person sitting in the control room.

When finishing the what-part of the analysis, it's time to focus on the how-part. This part involves construction of the operators user interface, construction of the human machine interaction, software and hardware. After the analysis part is finished the system needs to be tested. This phase is called the experimental part. These parts, will continuously alter between each other, as one see need for improving the system.

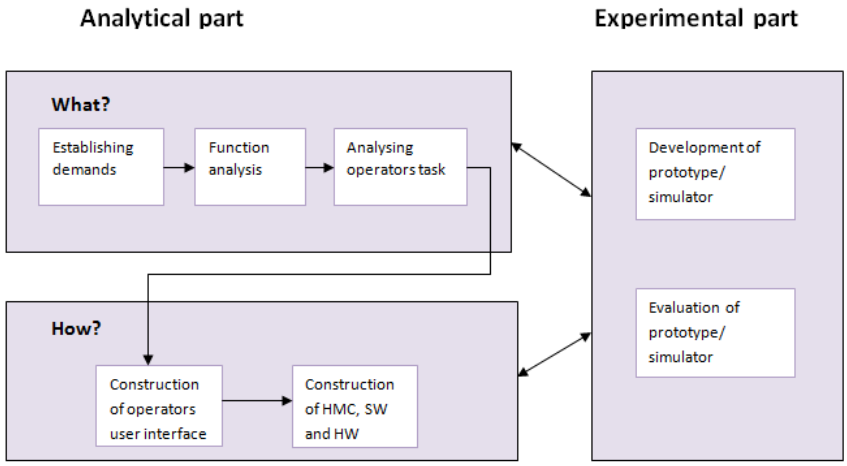


Figure 2.1: Bråthen's analysis of a system from lecture slides[1]

When automating a system there is several arguments for whether or not the system should be automated. While automation indisputably makes most processes faster and in some cases less dangerous than it has been earlier, meaning saving money, there is some downfalls as well. According to Bland[17] there is several reasons for that a system should not be fully automated. He mentions amongst others that jobs will be lost, as well as valuable information that earlier was handed down form generation to generation. The argument that is most relevant for this project was stated by Aune[18]. This argument is that when the level of automation rises, the level of attention the operator devotes to the system is smaller. The self-worth the operator feels also drops as the level of automation rises. This is explained further in section 2.2

2.2 Psychology and Mental Models

The commonly known Murphy's Law states "Anything that can go wrong will go wrong". Recently the American Dialect Society found an early version of the law, before it was generalized, stated by Alfred Holt in 1877, during a meeting of an engineering society.

> "It is found that anything that can go wrong at sea generally does go wrong sooner or later, so it is not to be wondered that owners prefer the safe to the scientific. It is also found that it is almost as bad to have too many parts as too few; that arrangements which are for exceptional and occasional use are rarely available when wanted, and have the disadvantage of requiring additional care. Their very presence, too, seems in effect to indispose the engineer to attend to essentials. Sufficient stress can hardly be laid on the advantages of simplicity. The human factor cannot be safely neglected in planning machinery. If attention is to be obtained, the engine must be such that the engineer will be disposed to attend to it."[19]

When applying this quote to the discipline of human-machine interaction as of today, it can be read as a warning of no matter how many functions one add to the system, one also need to consider the rambling mind of the human that is going to operate it. Most of the accidents that happen in a large system is a human mistake, according to Onshus [20]. Humans have a tendency to make short-cuts to avoid apparently unintelligent and thorough procedures. When operating a Human-Machine system the likeliness of a mistake with the system is bigger than a mistake on the actual hardware.

Human mistakes could appear in any phase of the lifespan of the technical device, and it is important to reduce the number of these mistakes to a minimum. According to Johnson[20] 85% of work-accidents happen because of uncertain actions made by humans. This number is debated, with some of the main arguments being:

- Fault for accidents is based on that the operator did not manage to prevent the accident, not that they were the cause of it.
- The operator must often interfere when the system is already on the borderline of its constraints. Then the operator must diagnose and put to work the correct solution. This is almost always done in situations with not fully known conditions and therefore time pressure is higher than for a regular job, and with little knowledge of what's wrong.
- To distinguish between operator and system faults is hard. The operator is often dependent on the design and procedures already made, and a bad design will not give the operator enough support in a pressured situation.

Stated by Bråthen[16] the different levels of automation and task entropy is important for the effectiveness and how the operator would feel about him-/herselves when working. This is illustrated in figure 2.2. The more predictable the work is, the less valued the operators feel when doing their work.

In addition it is easy for a person to loose interest in the job if it is not really a job worth doing. Sheridan [21] splits the different levels of automation into the following categorizes:

HIGH

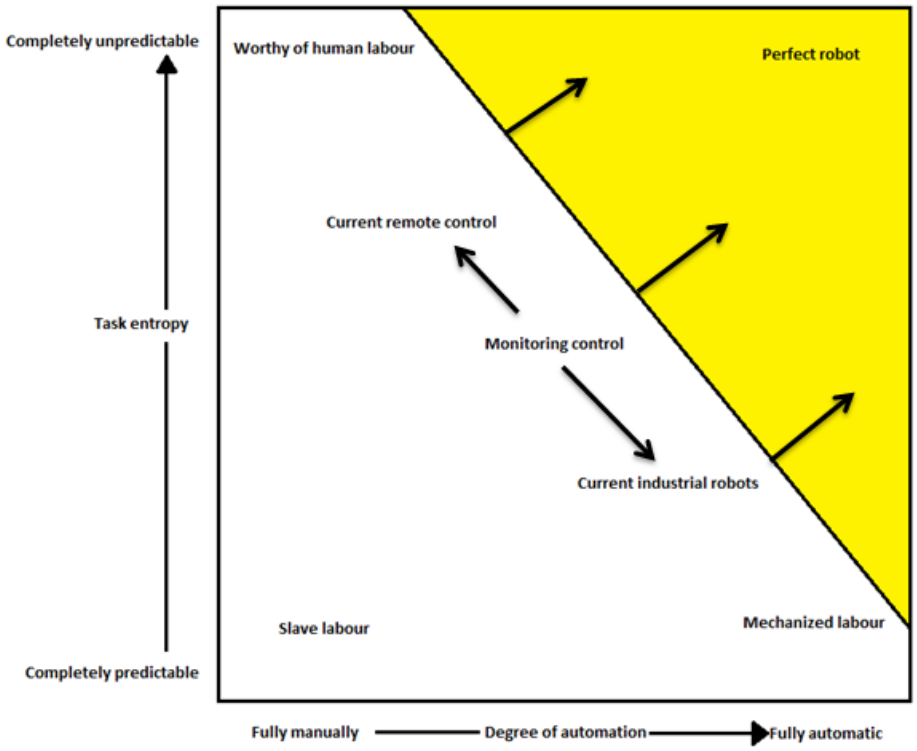


Figure 2.2: Correlation between task entropy and degree of automation

1. the computer decides everything, acts autonomously, ignoring the human
2. informs the human only if it, the computer decides to
3. informs the human only if asked
4. executes automatically, then necessarily informs the human
5. allows the human a restricted time to veto before automatic execution
6. executes that suggestion if the human approves
7. suggest one alternative
8. narrows down the selection to a few
9. the computer offers a complete set of decision/action alternatives
10. the computer offers no assistance: human must take all decisions and actions.

LOW

The higher the level of automation is the less work is left for the operator. This level of automation is not wanted in a system where it is important that no mistakes happen.

Rasmussen's model for the operator behaviour will, as seen in figure 2.3, explain how the operator perceive information and on what kind of level the information will be processed. It is important to remember that normally the operator will change where in this model one is operating. Starting with the job for the first time, one will work at the rule based or knowledge based level, while when one has done a task multiple times one will move this task to the skill based set of knowledge. The knowledge based actions is more demanding than the two others, and involves that the operator would need to have more understanding and insight of the system, than required in the other two. It is desired that the operators will move from one level to another as they work.

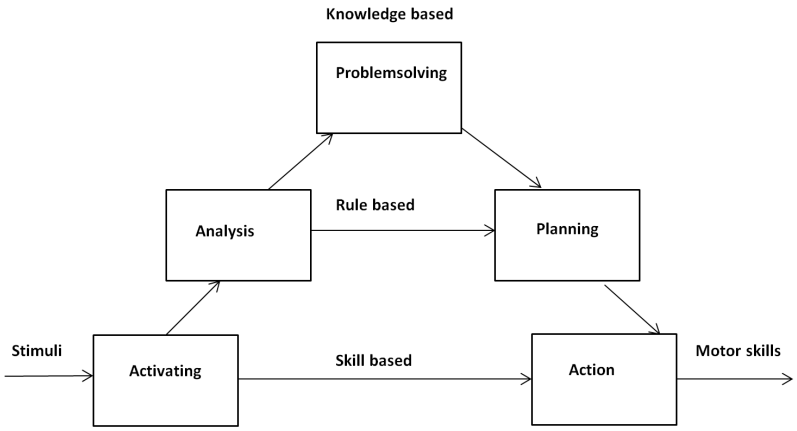


Figure 2.3: Rasmussen's model for operator acting

One of the main reasons for mistakes in a system is because of stress level. Typical reasons for stress can be stated as following according to Onshus[22]:

- Risc.
- Maladjustment of relationship between current threat and available measures.
- Time pressure.
- Conflicting alternatives (such as between economics and safety).

The relationship between performance and amount of stress can be seen in figure 2.4. If the stress level is to low it might lead to reduced vigilance, while to much

can lead to a feeling of powerlessness.

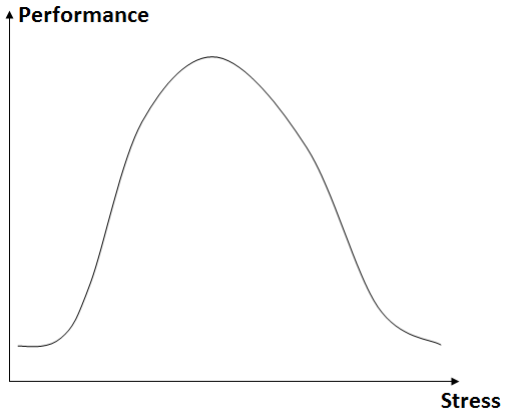


Figure 2.4: Correlation of Performance and Stress

2.3 Power Plant Communication

The power plant industry is an industry that is extremely concerned with safety, and is therefore one of the industries that implement new technology latest of all industries. There is many reasons for this, but the main reason is explained to be that new technology makes people insecure and one is cautious about mistakes when the consequences is rather severe. This can also be seen in section 2.2. When the level of risk is high one will feel a higher level of stress.

As of now the communication within a power plant, is done by landlines. In this section there will be some information on how the communication happens in a control room and between the control room and field operators.

When Cain and Sun[23] wrote the article "Computer Applications for Control Room Operator Support in Nuclear Power Plants" in 1991 they surely did not anticipate the amount of computers that once were to take place in the year of 2013. In this article they debate that "utilities are increasingly looking for ways to improve overall plant performance, rather than pursuing those safety-only strategies which have little impact on reliable day-to-day operations". They also mean that the renewal of the control rooms will need to fulfil at least one of the following criteria:

- decrease operations and maintenance costs and increase people productivity
- increase of production and decrease outages

- improve safety and reduction of safety challenges

Thus an upgrading requires strict cost benefit justification.

In 2006 De Carvalho[24] wrote an article about ergonomic field studies in a nuclear power plant control room. In this article the author wrote about how the operators would use verbal communication to maintain use verbal exchanges to produce ceaseless and recursive interactions, which would partake ensuring that the mutual and individual awareness was maintained and continuously updated. The communication is important to discover and avoid errors and faults, and by that obtaining the stability and safety of the system.

The oral communication between the control room and the plant plays a fundamental role for the Nuclear Power Plant (NPP) operation and control. Field operators, maintenance, instrumentation and engineering people are in constant contact with the operators in the control room. As written earlier the most common way of communication within the plant is by telephone, but there is also speakers spread around the plant as well as an intercommunication system that connects the control to some specific areas of the plant.

When the control room want to reach the field operators a call is made at the general broadcast system and the field operator in question will have to go to the nearest intercommunication cabinet and place a call from there. Previously radio and pagers are not aloud inside the plane, because of the high risk of electronic interference with the integrated circuits. Carvalho[24] made studies on this subject and they observed some difficulties in this routine. One of them is that there is several so called "blind spots" in the plant, where there was no speaker available or it was to noisy to hear the sound from them. Another was that the field operator had to stop his work to answer the call from the control room.

The same author wrote an article some years later [2], were it was explained how the communication between the supervisors in the control room and the field operators were executed. This is displayed in figure 2.5. As pictured the control room will observe an alarm, a fault or something else. When this is discovered they find the correct procedure to be carried out as a cause of this. Inside the control room it is procedures of most of what is going to be done. If the control room operator is insecure about the information on the screen or that the procedure that should be carried out is outside his/hers reach, a broadcast will be made to the plant. Each field operator is responsible for an area, and the responsible operator will be called. Example of the communication between the field operator and the control room is displayed in appendix A. As this paper only is concerned with the communication between field operators and control room, the communication inside the control room will not be explained. Carvalho[2] points out here as well that the field operator will need to take a pause in his/hers current task to talk with the control room.

An internal paper from IFE[25] states that correct communication is defined as that operators in the control room shall have such a way of working that

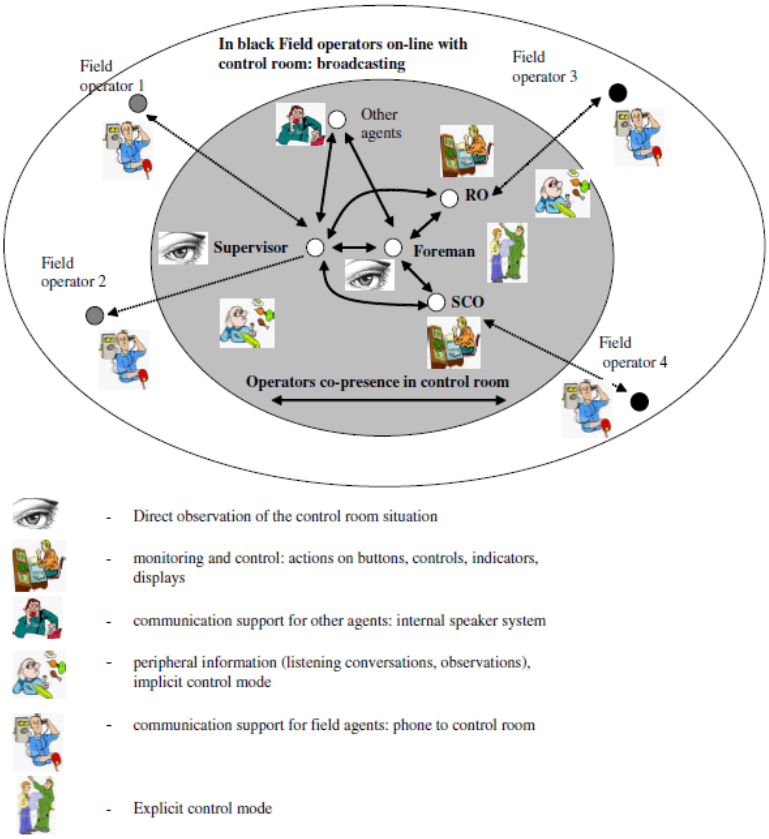


Figure 2.5: Communication in a Nuclear Power Plant [2]

when the shifts change the information should be exchanged in a manner that everybody is informed of everything that happens. The paper further defines two types of messages; informational messages and orders. The difference is that no actions are required after an informational message; this is sent to per example describe the process current or future state. An order is sent when something is supposed to be done. Accidents have occurred because of failed communication, and in some cases the failures were that it was unclear whether the information was an order or not. It is in other word important to make it clear which type of communication the current message is.

Shortcoming of communication is a major reason behind spreading of rumours and negative speculations. It can in some cases lower the effectiveness in a group and the motivation of each individual, as one is forced to guess what is going on in the group. During ongoing operational work there is need for frequent information of the status of the process. Partly to synchronise the perception the group have of the process as well as to mentally prepare them for upcoming

Order	
Name	The address of the order. Can be left out if the contact is already is established.
Background	To motivate and avoid mistakes. Can be left out if the recipient is already familiar with the circumstances.
Measure	Verb
Object	Per example system number and component number. In cases where well known terms, such as reactor tank or feed water tank, these can be used.
Possible alarm	Personal security or operation
How to report back	Make clear how the debriefing is supposed to happen. Can be left out for orders that is carried out immediately, and where sender and receiver is in constant contact with each other.
Receipt	
Measure	Repeat content from received order
Object	Repeat content from received order
Warning	Repeat content from received order
Report back	Repeat content from received order
Debriefing	
	This item should contain that the measurement was completed and to what degree it was finished.

Table 2.1: Form for order and receipt

events and notifications. The sender of the information is responsible for getting a confirmation of the recipients that the messages are received. This can be done in several was, eye contact, a nod, an affirmation (per example "ok"), or that the receiver repeats the message. The template for the order and the receipt is displayed in table 2.1.

2.4 Previous Chat Experiences

Previously there have not been any published papers regarding chat applications or synchronous messaging in a power plant environment. However, there has been some development of chat applications in similar environment. This section will contain information on that subject.

The word chat is defined by the Oxford dictionary as "the online exchange of messages in real time with one or more simultaneous users of a computer network"[26].

Gonzalez [27] defines chat as "synchronous communicative spaces which are incorporated into on-line activities" in her article, and Almeida d'Eca [28]defines chat as a "two-way synchronous form of computer mediated communication between two or more people by means of a computer". Synchronous communication refers to real time communication, interaction with live audiences.

Marrian-Webster [29] defines chat as "informal conversation, to talk in an informal or familiar manner", and that is what makes chat a natural space for communication to take place. The language in chat is usually composed of short phrases and a special language which makes communication closer to face-to-face conversation. Still chat that does not involve a video-stream or voice-stream will in the end have a higher risk of confusion, as body-language and voice supra segmental levels are missing.

An article written by Gonzalez[27] involve a paragraph that suggests that the participants get together in a chat to accomplish a real-life task. The activity should be planned and structured in such a manner that the involved parties always know what is supposed to happen. In this kind of chat, there is not necessary to have an moderator in a conversation, as the group will establish norms and handle situations as they go along. In some cases there will be important to keep an eye on the chat as there is some problems that could appear.

- People talk without respecting turns, not always, but it is very frequent that people start answering without waiting for the person who has the floor to finish.
- People introduces new topics without finishing previous ones.
- Taking turns is not usually well distributed. Some individuals tend to hold the floor or participate more than others.
- Some people only listen to the dialogues taking place.
- Different threads may be going on at the same time: two or 3 people are talking about something while others are pursuing some other topic (even if they are not next to each other).
- People attend to the thread that is of their interest, and may change their attention after a while, while some may participate in two or more different threads at the same time, which only depends on their ability to concentrate.

According to Handel and Herbsleb [3] it is potentially an important feature in a communicating program designed for workplace that the system supports both synchronous and asynchronous communication. Spoken communication can, like synchronous messaging and contrary to emails, change direction in a manner of seconds. But it is still possible for other parties that join the conversation on a later time to catch up on the content of prior information exchanges. Even

though other may choose not to get involved in the conversation they still will posses the same information.

2.4.1 Teaching Purposes

Let's Chat

A project done with chat programmes used for teaching foreign language is written about in an article by Stewart and File [30]. They developed a program called Let's Chat. In this article the main goal was to develop a project where one could chat using exclusively pre-stored utterances.

In advance the developers had sat down with the learners of the language and decided which phrases would be well suited for implementation. They divided the utterances into six categories (introductions, family, friends, hobbies, holidays and studies), where the learners were to communicate with other learners. All the conversations where observed by an evaluator, who scored them across three key measures of linguistic competence (command of vocabulary and idiom, clarity of communication, fluency) using a five-point Likert scale (1 = very poor, 5 = excellent). The resulting conversations were then ranked according to their respective aggregate scores and the results analysed. The system sketch for this communication tool may be seen in figure 2.6.

For three of four groups, all conversation using the system with the pre-stored utterances was ranked higher than the natural interactions. In the remaining group, the first and second place of the best conversations was taken by the system with the pre-stored utterances, but the third one was taken by natural interaction. When comparing all results, the results from the computer aided communication received significantly better results. The mean aggregate score of the computer-aided conversations was 11.75, while the mean for natural conversations was 8.33.

Avoiding Chat Confusion

A group of scientists from Rio de Janeiro developed a chat tool to be used in educational debates. As one of the main goals in a debate is to avoid confusion, this was one of their main concerns when making the chat program[31]. One of the first problems that occurred was that users sometimes lost track of which message was currently answered. They decided that the reason behind this was because of the non-linearity of the chat session messages. A HyperDialogue tool was then launched, where the sender marked which theme s/he was responding to. The drawback of this was that 7.5% of the messages was wrongly marked and made the conversation even harder.

Another problem mentioned by the users was that it was difficult to read all the messages during the debate. This problem was aggravated when many messages

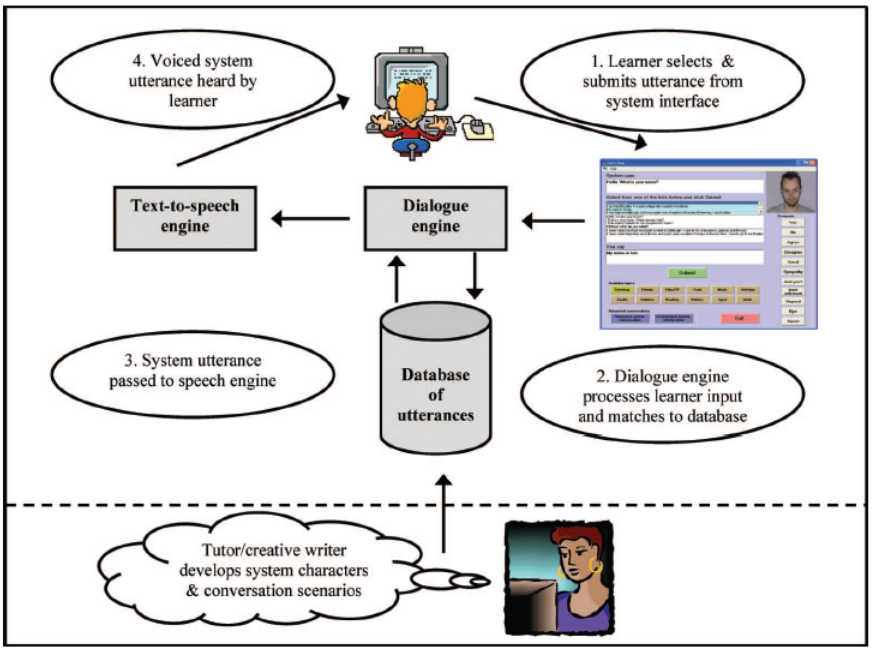


Figure 2.6: System figure of chat program

were exchanged within a short period of time. They solved this problem with a timer, which added a delay between each message. This solution did prevent the users having a feeling of information overload, but some complained that they felt it was slow, and that it was difficult to respond without knowing all the details of the prior conversation.

The final improvement the programmers did was to easier display information about the message. This was mainly a visual improvement as to not display the full name, only the nickname. Even though the messages have to have main focus, the other information was easily seen, and messages from the system about who left or entered the conversation were set to another display.

2.4.2 Air Control

In air-traffic control written messaging has been preferred as a mean of communication for a time. Rognin[32] wrote an article called "Human Communication, Mutual Awareness and System Dependability. Lessons learnt from air-traffic control field studies." In this article they define communication to be "every attempt to made by one person to distribute or/and acquire information. This involves both the production and the reception of the messages."

Communication in air control constitutes of both verbal, written and gesture based communication. The physical behaviour, para-verbal signs and environmental resources are used to give information or to acquire other's attention. In the air control situation addressed messages are sent intentionally to one or several receivers, who may be designated more or less explicitly according to the context and to the communication media. A message may explicitly mention the intended receivers, by their name or the identification of their role.

There is not a synchronous way of messaging each other, but the messages are still possible to be delivered as synchronous messages when the attention is achieved.

Contrary to messages where the intended receivers are mentioned, the non-addressed communication corresponds to messages that are available to the receiver without the explicit intention of the sender to send them these messages. This would be due to communication support. This does not mean that the sender is not aware of the fact that messages are available to these non-intended receivers. Neither does this mean that the sender does not want them to receive the message.

This article [32] concludes with that cooperation in a shared workspace with between operators with closely related tasks and skills, naturally exhibits powerful dependability features, where redundancy and diversification are exploited within the team as means for preventing or even tolerating potential errors from team members. These capacities are based on several factors, such as mutual awareness, which is mainly supported by human communication within the working groups. This includes the underlying mechanisms associated to human communication to improve its efficiency and dependability.

2.4.3 Social Messages in Workplace

Misunderstandings during synchronous messaging in a social environment do not have as severe consequences as when using the same technology in the industry. It is however some concerns that needed to be taken.

One of them is the loss of concentration. The addition of alarms that pop up on screen at any moment when receiving a message, with some of them being unfortunate, are destroying workers concentration. If these messages in addition are of a personal nature and not of professional interest, there are some other aspects to be considered as well. Slatella[33] points out that the factor of who gets a message and who does not get it, is a pointer of social cliques, and is similar to the pecking order found at some high schools. This again can cause a feeling of not belonging, and insecurity in the workplace, which again can affect the performance of the workers. The article by Handel and Herbsleb[3] point out that users that are not familiar with instant messages reported that group chat tended to be less distracting as one-on-one messages. This was because when a message arrived just for one person that person felt obliged to respond.

If the message came to an entire group there was several users able to answer and one would not bother doing it if it disturbed ones own work.

Another downfall of communication inside a workplace is the content of non-work messaging. The article written by Handel and Herbsleb displays the picture in figure 2.7. In this article they write that during the day, the number of work related messages sent during each thirty minute interval is rather consistent during the day. However the number of non-work and humour messages increase significantly when one is closing in on the end of the work day. Which result in a rather steep boost in the percentage of non-work messages displayed in the curve. The most research of social messages in a workplace takes place in

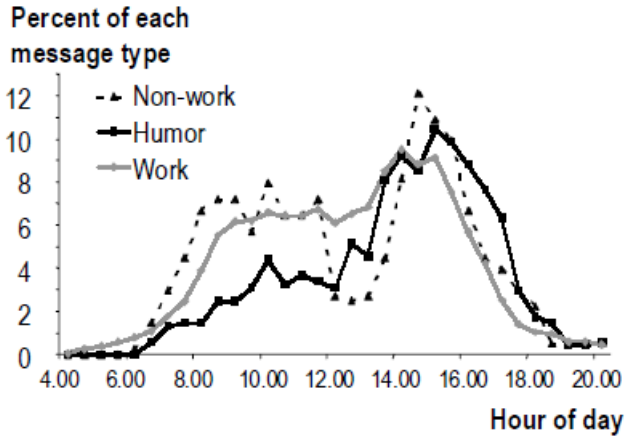


Figure 2.7: Percentage of Non-work, Humor and Work messages that occur on average in each 30-minute period of a weekday[3]

an office environment. There are a lot of similarities between social software in offices and in industry, but one main difference is that work in a process plant takes place in shifts, and the continuous production requires real-time monitoring. Control rooms are equipped with several monitors, and computer systems. In addition to watch over the steer the production the personnel in the control room need to collaborate with other areas of the production as well as experts off-site.

Social software in an in industrial context should be closely integrated to the existing work context and existing ICT-infrastructure such as: production systems, product data management, customer relation management, and other enterprise resource planning software[34]. The link between these systems and new social software is largely missing as of the time the article was written. However the author believes that the most likely solution is to wait for the next generation of system and include the social software in them.

Koskinen[34] made a test software that were going to be used within two dis-

tributed collaboration networks; between a group of experts centres and between an expert centre and a process plant. The software was used for a year, and in that time 220 messages was sent. The usage was not obligatory, and about 30% of the workers had used it at some point. There was not noticeable reluctance of using the software, event though it was unknown. It seamed as everybody was rather comfortable asking questions as well.

2.5 Commonly Known Android Communication Applications

In this section there will be presented several commonly known chat applications available for an Android unit. The applications is selected from Google Stores most downloaded applications.

Many of the applications have obtainable applications for other units, and there is also some communication applications that is only available for iOS-users. Neither of these options will be considered in this report.

2.5.1 Facebook

Facebook is an online social networking site, which as of March 2013 had 1.15 billion active users [35]. This site requires that the users register before using the site, and with that opening a profile. From this profile one can add other users as friends. It is then possible to communicate with these friends in the chat that is included into the website.

The Facebook application for Android was launched in 2009, and received generally bad critics. However the application is, as of 17th of November 2013, the second most downloaded application in Google Store. The application consist of several parts, such as a newsfeed with updates of so-called statuses from friends, as well as a chat-application as shown in figure 2.8. This chat-part consist of one view with a display of the chat itselves as seen in 2.8a. A list of currently available users is accessible when dragging a finger from the right side of the screen towards the left side, seen in figure 2.8b. An overview of conversations one has taken part in is also available, seen in 2.8c.

A close-up of the chat it self is shown in figure 2.9. The layout of the chat consists of speech bubbles, where the messages are displayed. The users are separated by gravity of the speech bubble, as the ones sent from ones own unit are placed to the right, and received messages is placed to the left. In addition to the gravity the colour is set to different colour if one is sending or receiving the message. To see who the message is from it is added user pictures in front of the message. In this application they have added the possibility of adding so called stickers. These stickers are pictures meant to display a feeling.

(a) Chat (b) Online users (c) List of conversations

Figure 2.8: Screen shot from Facbook Android application[4]

The layout bar on the bottom of the application consists of four items;

- A plus-sign that uploads picture or videos stored at the unit
- A field to write the message in
- A smiley-button to send stickers
- A Send button

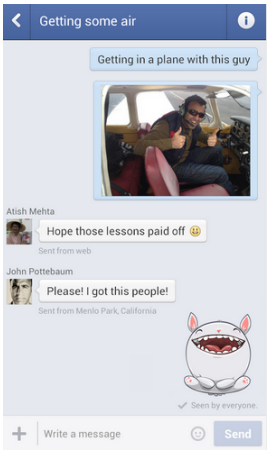


Figure 2.9: Screen shot from Facbook chat[5]

2.5.2 Viber

Viber is a proprietary cross-platform instant messaging Voice Over Internet Protocol (VOIP) application for smart phones, developed by Viber Media[36]. The application was initially launched in December 2010 for iPhone, and for Android in May 2011. As of may 7th 2013 the application had 200 million users. Viber is as of December 8, 2013, alongside Skype, the most common communication application for Android invented outside of the US. Contrary to many other communication applications Viber is not based on an internet-account at their website, but on the users' phone-number. This phone-number is used as a user name and if a newly joined user matches a number in the telephones phone book, a notification will be sent to the user having the phone number in his/hers contact list.

The functionality of the application contains the normal chat-functionality, as seen in figure 2.10a. One can add pictures and there is a large collection of sticker that can easily be accessed by pressing the pluss-sign. The contact list in figure 2.10b is also rather similar to the other applications discussed in this section. The thing that makes Viber differ from the other applications in this section is the ability to send video- or sound-files via the chat view. To do so one press the pluss-sign and select the option one want to. The sound-files are not available in other widely distributed applications.

(a) Chat (b) Contact List (c) Sound-conversation

Figure 2.10: Screen shot from Viber Android application[6]

Upon receiving a SMS from a contact on Viber this message will also open in

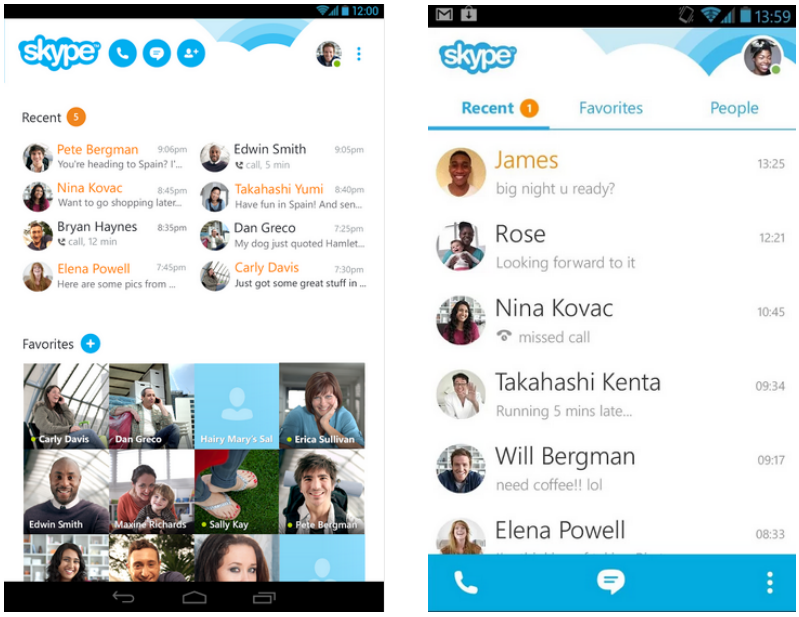
the application. The drawback is that it is not possible to send SMSs from the application, and one the need to open the message program one selves and send a message from there.

The layout in the chat application separates the messages in a similar way as facebook 2.5.1, with different gravity, colour and user pictures. It is also possible to change the background of which the messages are displayed on. The bar on the bottom consists of the following items:

- A plus sign to add stickers.
- A field to write the message.
- A button to turn on/off the sharing of ones current position.
- A microphone button that makes it possible to upload the sound file

2.5.3 Skype

Skype is a VOIP and instant messaging client developed by the Microsoft Skype Division[37]. The program was first launched as a desktop program in August 2003. As of June 2012 Skype had passed 70 million downloads on Android devices and had 34% of the international call market share.



(a) Recent activity (b) List of conversations

Figure 2.11: Screen shot from Skype Android application 1[7]

The service allows users to communicate with their contacts via voice, video or instant messaging. An option would also be to combine these. Calls to other users within the Skype system is free of charge, while calls to landlines and cellphones via traditional telephone networks cost money. This is withdrawn from an account with deposited money. This application also allows file transfer and video conferencing. To use Skype as a company, video conferences cost money.

The Skype Android application contains a list of conversations, as shown in figure 2.11b, and the chat it self as shown in figure 2.12a. Also in this application it is possible to participate in group conversations. Figure 2.12b displays the User Interface when calling other users. The displayed picture is a call without video, and therefore the profile picture is used. Skype also have a home page with the recent status updates and events from contacts as well as the people that is called the most often 2.11a.

(a) Chat (b) Call

Figure 2.12: Screen shot from Skype Android application 2[7]

Contrary to Facebook in section 2.5.1 and Viber in section 2.5.2 there is no speech bubbles seperating the different chat entries. The only thing seperating is the colour of the entry as well as the picture of the sender. The menu bar in this application contain two menu bars with the content:

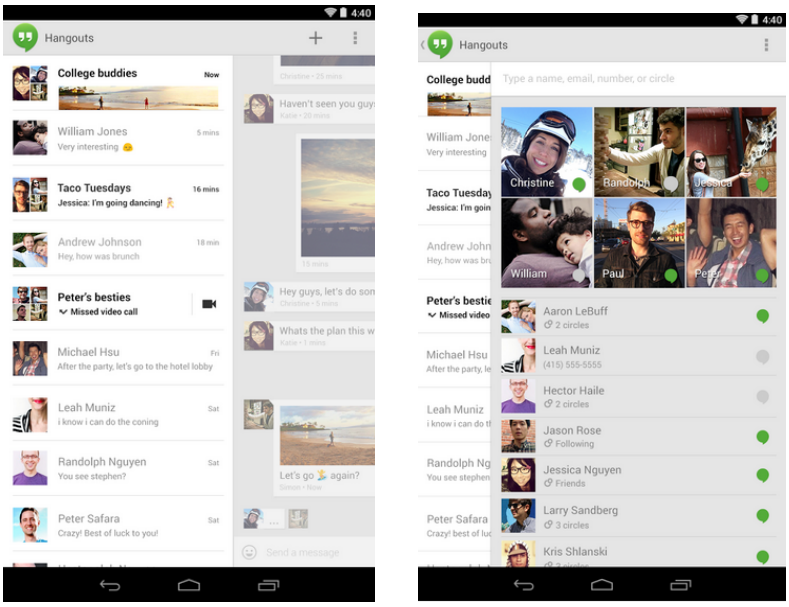
- Upper menu bar

  - A plus button thet upload video, files of pictures.
  - A area to write the message
  - A smiley button to upload smileys
- Lower menu bar
  - Video button to start video call with the person included in the current conversation
  - Phone button to start a voice call to the person/s included in the conversation
  - Add a person to the conversation
  - A menu bar that launces further options

2.5.4 Google Hangouts(Earlier known as Google Talk)

Google Hangouts is an instant messaging and video chat platform developed by Google, and launched on May 15th, 2013. From before Google had several messaging products, namely Talk, Google+ Messenger and Hangouts, a video chat system present within Google+. This application is standard on most Android phones as one need a google account to access the Android applications in Google Store. The list of contacts, as shown in figure 2.13b, is generated through the users gmail contacts, that also have gmail accounts. The contacts most frequently contacted persons appear at the top as favourites. The list of recent conversations can be seen in figure 2.13a. The chat-view it self can be seen in figure 2.14a. In this application it is possible to send standard text, maps, pictures and emoticons. Emoticons is pictures of a very small size, that can be included into the text for entertainment of extended information. One could chat with one person or start a group chat as one pleases. The same applies to video calls, seen in figure 2.14b. Google Hangouts have a rather typical layout, similar to those in Facebook and Viber. The classic speech bubbles are there, only in Google Hangouts they are square and not rounded in the corners. The menu bar consists of the following items:

- A smiley button to add smileys or emoticons to the conversation.
- A area to add messages
- A way to turn on/off the location of ones wherabouts
- A camerabutton to upload pictures



(a) List of conversations (b) Online Users

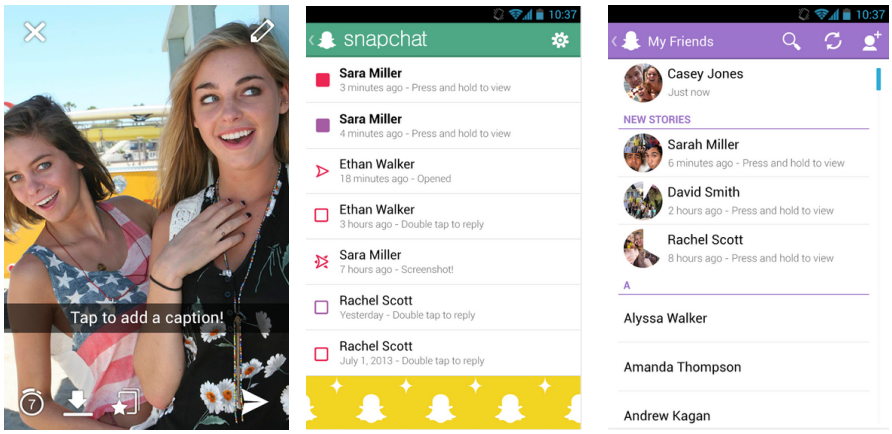
Figure 2.13: Screen shot from Hangouts Android application 2[8]

(a) Chat (b) Call with several users

Figure 2.14: Screen shot from Hangouts Android application 1[8]

2.5.5 Snapchat

Snapchat is a photo messaging application developed by Stanford University Students[38]. When using the application users can take photos or record videos and send to their friends. Before sending the user could add text or drawing onto the picture as seen in 2.15a. These photo/video-messages could be sent to a list of pre-approved friends, which can only see the message for a certain amount of time, ranging from 1 to 10 seconds, after they have opened it for the first time. If one of the recipients take a screen shot of the picture, a notification is sent to the sender, as seen in the list of messages shown in figure 2.15b. The list of previously approved contacts is seen in figure 2.15c. The application was initially released in September 2011, and as of June 2013 had 8 billion adult users in the US alone, and an average of 200 million messages exchanged each day[39]. There is however no available data about the total amount of registered users.



(a) Message (b) List of messages (c) List of contacts

Figure 2.15: Screen shot from Snapchat Android application[9]

In September 2013 Forbes published an article revealing that the snaps was not deleted from Android phones after the time lapsed, although this was claimed by the Snapchat developers. Instead they were protected by a ".noMedia" file extension, and therefore kept from the user. This led to a media storm with warnings about not using the application. As the amount of users is not known it is unsure whether this article influenced the users.

Chapter 3

System Suggestion

In section 2.1 one can see Bråthen's suggestion for how one is supposed to implement a system. It exists of three parts. The first one is a analytical part, which contain three items; Establishing demands, Function analysis, and analysing operator tasks. For readability some of the theory will be discussed in this section.

The first thing that needs to be done is to establish demands. The essential demand in this situation is the need to utilize new technology in industry. As stated in section 2.3 there is three main concerns when adding new technology to the Nuclear Power Industry. To take advantage of new technology can potentially fulfil all three of these conditions if it is done properly. To add a chat system can decrease operations with the fact that the field operators would not need to walk to the phone for simple messages, and therefore increase the productivity. When increasing productivity one can possibly also increase production. The safety can be improved with the continuous messages if something is wrong.

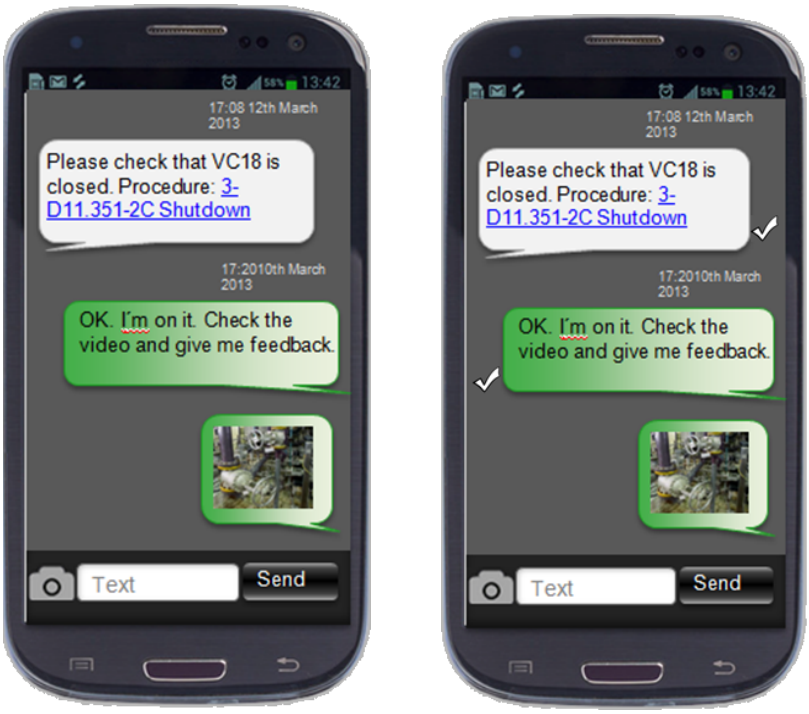
Function analysis is the second item in the list. With this part of the system Bråthen mean that the system designer should think about the different problems that can occur, and which measures should be taken. This will be done subsequently in this chapter. Analysing operator tasks would in this case involve finding out what the field operator does. As this assignment is only considering the chat-part of the application this would mean that the field operator's task is to do his tasks, and then report the results to the control room.

From Bråthen there is two parts of the system analysis, and the second is the how-part. This part consists of construction of the operators user interface, and construction of hardware and software.

The hardware is already determined, and the device this application was going to be made for was a Samsung Galaxy SIII. The device it self is 136.6 x 70.6 x 8.6

mm, and with a screen size of 720x1280 pixels. This screen is rather large for a cell phone these days, although there is several tablets that would have a larger screen. A larger display would add the benefit of better overview of the system, but one would loose the advantage of being able to carry it easily everywhere. It would be a fitting phone also because the battery time is estimated to be 50 hours. This is more than sufficient for a shift. The telephones specifications can be seen in appendix B.

Before starting making the system, a presentation of the system in total was taken into consideration. Some of this system can be seen in appendix E, and the rest is classified. In the end a suggestion for the chat layout was shown. This layout was can be seen in figure 3.1a, and was the base of the layout for the application. This chat-application is meant to be an addition to the current broadcast-landlines system, and not a replacement.



(a) Original chat (b) Read Confirmation

Figure 3.1: Chat functions, and read confirmation

The suggested layout would look somewhat alike the layout of the facebook-application shown in section 2.5.1. It contains the characteristic speech-bubbles with the rounded corners, known originally from cartoons. From section 2.4.1 the articles stated that an important item in a manner of avoiding chat confusion was to make sure that everybody knew at all times who sent which message. In

the system suggestion this was based on the gravity of the speech-bubbles, in addition to the colour of them. In section 2.5.2 there was added an opportunity of changing the colour of the background. It is also common for chat-applications to have the possibility of sending drawings. Neither of these items seamed necessary to add in an application used purely for communication between the operators in the control room and the field operators. The original suggestion had also added the received time of the message.

In section 2.5 one can see that most of the application has the possibility of sending multimedia messages. This would consist of pictures (Facebook, Viber, Google Hangouts, SnapChat), video conversations (Google Hangouts, Skype), phone-conversations (Viber, Skype) and sound-files (Viber). Skype also have the ability of transferring files. It was suggested in the system suggestion from IFE that one should add videos. For majority of the applications it is sufficient with the ability of sending pictures, and it should be so for this application as well. The control room would in addition have constant supervision with the power plant on their screens it would not be as necessary to send them a video of what is going on in a plant. Even though the noise cancelling technology of the Samsung Galaxy SIII has been advertised for, it is not sure it is sufficient. If it cleans the noise that is sent to the plant, it is not sure that the field operator will hear anything if a VOIP was performed. This would lead to that the operator would have to relocate and then one could just as easily use the landlines.

One of the reasons behind confusion in a debate, such as in 2.4.1, was the amount of messages sent at the same time. The solution to this was to add a timer so that one had a certain amount of time for reading a message before a new one arrived. This would cause some annoyance that one would not at all have all the information currently available. In an application used in a NPP, some matters are more urgent than others. As seen in 2.3 there is parts of the plant that is not reachable by the broadcast-system. The possibility of having unimportant messages blocking the sending of higher prioritized messages would in the system suggestion seem as a larger drawback than some irritation of not having time to read all messages. In addition the exchange of messages would probably not be as frequent in a chat-system between a limited numbers of people, as in a debate environment.

When the layout of the main view of the chat was decided on, improvements were in order. From the section on power plant communication 2.3 it was stated the template of the communication between the control room and a field operator. In section it is stated that the sender of the information is responsible of getting an confirmation from the recipients that they would have gotten the information. When talking to someone face to face, this is usually done with a nod or eye-contact, and if one is talking to someone over the phone one can simply say "ok". A way to simulate this confirmation when communicating in writing is to add a so called read check, as seen in figure 3.1b. This would work as a message to the sender that the receiver have opened the application, or program, and have seen the message. If the sender of the message then notices

that none have seen the message, one is able to call them and repeat the message over telephone.

Talking to someone face to face will leave a lot of the conversation to body language. Some of this can be translated into a phone call, with the sound and intensity of the voice. All of this is however lost when talking through writing. A way the chat-applications from section 2.5 has solved this is to be able to send emoticons. As this seems to be kind of unnecessary in a professional situation, this was not done in this application. However to simulate a simple nod of the head the read confirmation was implemented. This function needed to be implemented in both the desktop program as well as in the portable device.

Another improvement to the chat program from 2.4.1 was to add tags to the message, which explained which item the information was about. From section 2.3, it was explained that each plant-component in a NPP have its own identification number.

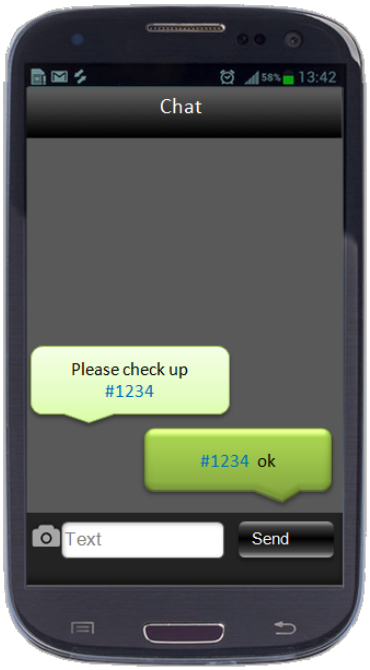


Figure 3.2: Log of messages

A symbol that is not often used in any language is the symbol #. In several social networks, for example Twitter or Instagram, this is used as a hashtag. Hashtag is a form of metadata tag, which makes it possible to group the messages together. When adding this symbol in front of a number, the following word will transform into a link, as seen in figure 3.2.

When pressing the number, as seen in figure 3.3a, the view in figure 3.3b would

appear. This view would display every message that has a link to the current item being pressed.

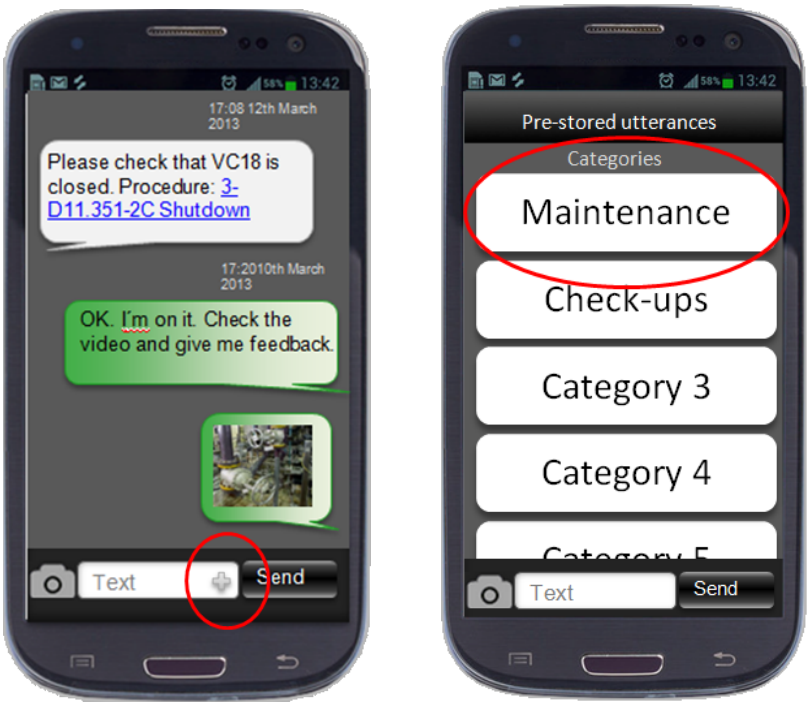
(a) Pressing link

(b) Display items with corresponding id

Figure 3.3: Pressing id, displaying list of messages

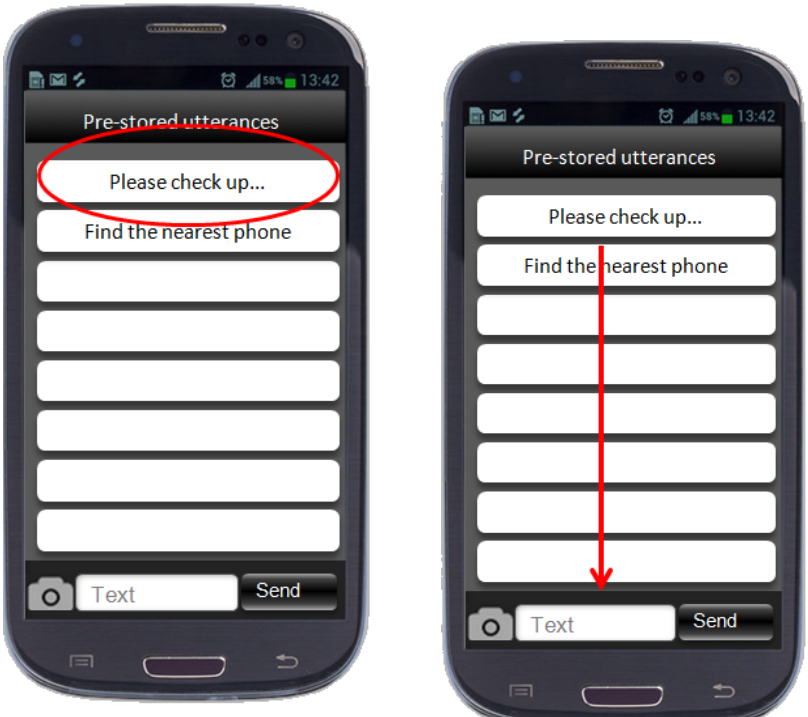
From applications made for teaching purposes seen in section 2.4.1, one can see that conversations that consist of previously stored utterances have, in average, higher rate of understanding and quality. Therefore improvement number 2 was suggested. As seen in figure 3.4a a plus-sign was planned to be added inside the text-field. This plus sign was then planned to be opening a menu of categories that contained the specific phrase. The need for categories instead of simply opening a list of phrases was justified with the amount of phrases needed in a conversation within a plant. To make the content more straightforward the requirement of categories was there.

When pressing one of the categories, as shown in figure 3.4b a list of category content would appear, where each row is selectable. After selecting an item, as seen in figure 3.5a, the content of the item would be added to the edit text field, as in figure 3.5b.



(a) New plus button (b) Saved Categories

Figure 3.4: New plus button, and Saved Categories



(a) Pressing button to add new phrase

(b) Press category

Figure 3.5: Phrase button and list of categories

This content would then be editable, so that it is possible to edit an entry. One can for example add a id, as seen in figure 3.6a. The definition of which phrases that would be the most fitting would vary from person to person. When being able to alter the phrase in this manner one can have the possibility of using the most fitting phrase and then change it to be exactly what one need. One can per example add a tag, to make the message sortable. After the message is completed one press the send button, and the message will be sent and added to the user interface, as seen in figure 3.6b.

(a) Add id to message (b) Added items to chat

Figure 3.6: Add id to message, and chat with phrase

The complete requirements specification for the application will be

- Making an application for an Samsung Galaxy SIII, with display size 720x1280 pixels
  - Adding a complete chat user interface with:
    - Speech bubbles, with message
    - Time of received/sent message
    - Some way of sorting who the sender is
    - Read confirmation

        * Button for starting an activity of adding saved content
        * Button for uploading and sending pictures
        * Area to write message
        * Send button
    - Activity to select between selected categories
    - Activity to select between phrases
- Making a desktop program, to received the messages from the applications. Only to be used in this project.

This system suggestion was presented to IFE and approved as the prototype to be implemented, and used as an input to the system tested during 2014.

Chapter 4

Implementation

This chapter contains information on how the application was implemented and what different parts it consist of. A suggestion of how to incorporate it into the already existing application is presented in E. There are some differences between the system suggestion in chapter 3 and the resulting application in this application. These changes are explained, and discussed in the discussion in chapter 6.

Included in this Master's Thesis, there is a CD with the source code. Appendix C contains the information on how to install the software to continue develpoment of the system.

A user manual for the completed system is also available in appendix D.

4.1 Adding New Projects

The systems need two parts. One program that runs on a computer in the control room and one program that runs on the portable unit. To do so, two projects was created in Eclipse. To implement the program used in the control room a Java project was created and for the program that was to be running on the portable unit an Android project was created.

4.1.1 Java Project for Control Room

20 years ago, one would need about 200 code lines to make a simple window in Windows. Now a days one can simply click the button **File->new->Java Project**.

This project will be the base for further development in the following sections. To run the Java project, select the "Run"-button available in the tool bar. The first time this is done a window will open. Choose "Java Application".

4.1.2 Android Project for Portable Device Application

Before starting implementing the Android project, one needed to do three things according to the Android Developer site [40].

1. Dowmload the Android SDK
2. Install the ADT plugin for Eclipse
3. Download the latest SDK tools and platforms using the SDK Manager

How to install these things is covered further in appendix C.

To add a new Android project click the button **File** -> **New** -> **Android Application Project**.. How to fill out the fields in the pop-up screen is displayed in appendix C, and is also available on the Android Developer site[41]. This could also be done with Command Line Tools.

The Android Project that is created is an example project that displays a screen with "Hello World". This project is the one the rest of the application is based on. When running the application one can choose between running the application on a virtual device or on a real Android device. As the real device was available during the programming process this was the option that was chosen. If that is not the case there is several virtual devices that resemble the Samsung Galaxy SIII available at the Samsung website[42]. These devices could be seen in the Android Virtual Device Manager, which should be available in the tool-bar if the instructions in appendix C is followed correctly.

To run the application, one needs to click the run button in the tool-bar. The first time this is done a window appears. To get a program runnable on a portable unit select "Android Application". Then a list of available devices show up, and one could choose the right one. The first time this is done it is also needed to enable the USB-development option on the cell-phone. When all this was done the system was ready for further implementation.

The folder structure in an Android project contain of several parts that follows when Eclipse makes the project. Table 4.1 displays a explanation of which folder that are included, and what they contain.

4.2 Simple TCP Server/Client

When the projects were created, the first thing needed to be implemented was a server/client system. This part of the program was going to take care of the

Folder	Sub-folder	Description
src		The source folder, containing the MainActivity and other Java code of the
gen		This folder contain java files generated by ADT. Also contain the special class R, which hold the references to various references placed in the application. All layout item have an id, which will be a part of this R-file, and used as links to that item.
assets		This is empty by default, and used to store raw asset files. This could be used to make the application work.
bin libs		Output directory of the build, This is where the final .apk file is placed.
res		This folder contain all the resources of the project, as images, layouts and values. Resources are non-code files that are used by the code and compiled in run-time.
	drawable	For bitmap files. The different drawable files is different Android units. The Samsung SIII use the folder drawable-xhdpi.
	layout	XML-files that are compiled into screen layouts.
	menu	For XML-files that define the menu of the application.
	values	For XML files that are compiles into many kind of resources. This folder can be the location of colours, strings, or other usable units. Not referenced by name, but by the id from the R-folder.
Android Manifest		The control file that expain the application, and each of its components. It defines the start-up activity, the target device, and permissions for the application.
ic_launcher-web.png		Application picture, displayed on the menu of the phone

Table 4.1: Folder Structure, Android Projects

packages that were sent from the control room to the Field Operator (FO).

Implementing a server/client system requires the use of a potocol. The main alternatives is either a TCP protocol or an UDP protocols, where the differences between the two can be seen in table 4.2. In the Nuclear Power Industry, two of the main concerns is dependability and security. This is important to maintain when choosing the type of protocol as well. As seen in the table the UDP protocol offers a connectionless protocol, while TCP is connection-oriented. When using the TCP-protocol, one is ensured that the data is delivered, with no corruptions, as long as the connection is active. The order of the messages delivered is also ensured to be in the correct order when using the TCP-protocol. This is not guaranteed when using a UDP-protocol. However the UDP-protocol is considered faster than the TCP. This is mostly because of the heavyweight. As written in table 4.2, the TCP-protocol require that if the low level parts of the TCP-stream arrives in the wrong order, a re-send command will be sent. In UDP, the network-card/OS does not need to do as much job with reassembling the data. However this was not considered important enough compared to the sorting and the connection-oriented benefits of the TCP-protocol.

This part is mainly implemented as a base for this application, and is probably not the one that is going to be used in future development of the application. When implementing this part of the program the inspiration was collected from the websites Java Code Geeks[43], Think Android[44], Android Solutions[45] and Edumobil[46].

4.2.1 Control Room

In this system both the control room and the field operator need to be able to send information to the other part of the program. Even so the program needed a server and a client side. The control room seemed to be the most fitting side for implementing the server part of the program.

Before starting to implement a Java file was added to the Java Project. In Eclipse this is done by right-clicking the source file in the project directory as seen in figure 4.1.

	TCP	UDP
Reliability	TCP is connection-oriented protocol. When a file or message send it will get delivered unless connections fails. If connection lost, the server will request the lost part. There is no corruption while transferring a message.	UDP is connectionless protocol. When you a send a data or message, you don't know if it'll get there, it could get lost on the way. There may be corruption while transferring a message.
Ordered	If you send two messages along a connection, one after the other, you know the first message will get there first. You don't have to worry about data arriving in the wrong order.	If you send two messages out, you don't know what order they'll arrive in i.e. no ordered
HeavyWeight	When the low level parts of the TCP "stream" arrive in the wrong order, resend requests have to be sent, and all the out of sequence parts have to be put back together, so requires a bit of work to piece together.	No ordering of messages, no tracking connections, etc. It's just fire and forget! This means it's a lot quicker, and the network card / OS have to do very little work to translate the data back from the packets.
Streaming	Data is read as a "stream," with nothing distinguishing where one packet ends and another begins. There may be multiple packets per read call.	Packets are sent individually and are guaranteed to be whole if they arrive. One packet per one read call.
Examples	World Wide Web (Apache TCP port 80), e-mail (SMTP TCP port 25 Postfix MTA), File Transfer Protocol (FTP port 21) and Secure Shell (OpenSSH port 22) etc.	Domain Name System (DNS UDP port 53), streaming media applications such as IPTV or movies, Voice over IP (VoIP), Trivial File Transfer Protocol (TFTP) and online multiplayer games etc.

Table 4.2: Some differences between TCP and UDP[12]

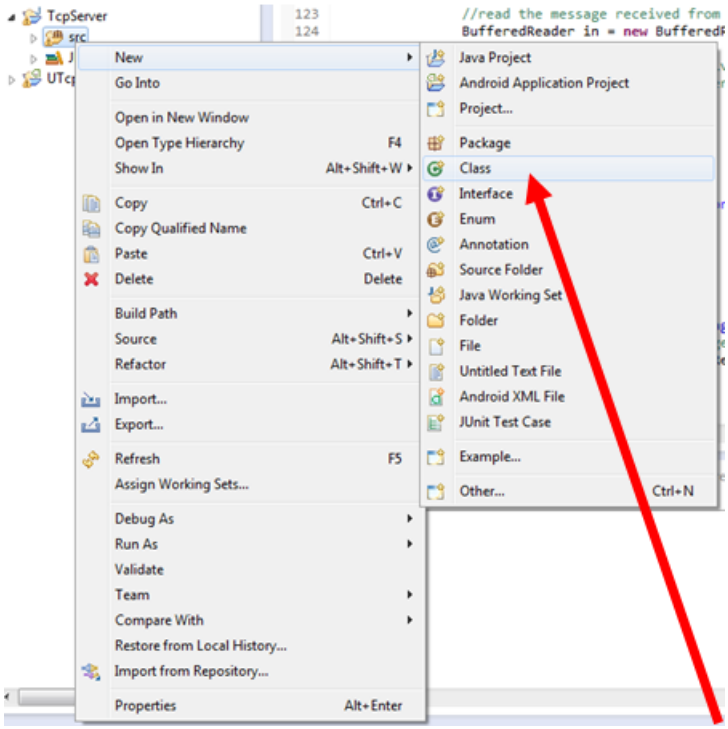


Figure 4.1: Adding new Jave File to Project

This file was where all the functions that were needed for the server side of the connection were placed. A port to send and receive on was also defined.

The functions placed in the server were as following:

- Constructor of the class:
  - public TcpServer(OnMessageReceived messageListener)
- Function where frame is to be added later:
  - public static void main(String[] args)
- Close the server
  - public void close()
- Methods to send the messages from server to client:
  - public void sendMessage(String message)
  - public boolean hasCommand(String message)
- Build a new server connection and listens for incoming messages
  - private void runServer()

4.2.2 Portable Device

As the control room was selected as the server side, the portable device was subsequently selected as the client side. In this case the TCP-functions that were needed were added to the file that Eclipse added when making the project, named "MainActivity".

An Android project is based on activities, where each activity is one screen on the device. Within these activities the human-machine interaction happens. When starting an activity a function called *onCreate()* is called. In this activity one specify what need to be done before interacting with the user.

To implement a continuous listener for received messages a thread was created. This thread would run continuously, alongside the activity on the device, and when receiving a message trigger a function in the activity it self. The thread can be initialized from the *onCreate()*-function, with the simple codeline *new Thread(new ClientThread()).start();*, with ClientThread being the name of the thread running alongside the TCP-listener. In this thread the port which the control room was going to send and receive on, as well as the IP-address of the computer that hosted the server was declared.

While the idea was to postpone the layout of the application until later, it was needed in this case to add a field to fill in text as well as a send-button to trigger an event of sending data over the TCP-connection.

Normally this is the place where layout is set, with the command *setContentView(Int id)*. The argument this takes is the location of a layout file where the user interface of the activity is defined in an xml-file. These ids are held in the gen folder, explained in table 4.1. It is most normal that each of the activities has it's own layout file, but it is possible to share.

The layout from section 4.1 consist of a simple Linear Layout, which is added to the file by Eclipse, when creating the example "Hello World"-project. When choosing the layout, there is several choices, as seen in table 4.3. Android Developers [47] suggest a simple way of adding a field where one could write as well as a "Send"-button. The suggestion is seen in figure 4.2. This suggestion consists of a LinearLayout with an EditText-field and a button, which is added to the layout file. As the goal of this part of the implementation was to send a simple message, this suggestion was the layout that was used. This layout can also be seen in many Android applications.

Figure 4.2: Simple EditText-field with Send-button

Type of Layout	Description
LinearLayout	Arranges its children in a single column or a single row
RelativeLayout	Position of children can be described in relation to each other or to the parent
TableLayout	Arranges its children into rows and columns
FrameLayout	Designed to block out an area on the screen to display a single item
GridLayout	A layout that places its children in a rectangular grid

Table 4.3: Android Layouts

When pressing the send-button a function called *onClick()* will be called. When this function is called, it reads what is in the EditText-field, and converts it to a string. How this is accomplished is displayed in listing 4.1.

Listing 4.1: Sending data with a PrintWriter

```
EditText et = (EditText) findViewById(R.id.EditText);
String str = et.getText().toString();
printWriter = new PrintWriter(new BufferedWriter(
new OutputStreamWriter(
socket.getOutputStream())),true);
printWriter.println(str);
```

And the xml-file which makes the layout is as in listing 4.2.

Listing 4.2: XML layout

```
<LinearLayout xmlns:android="http://schemas.android.com/
    apk/res/android"
     xmlns:tools="http://schemas.android.com/tools"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:orientation="horizontal">
     <EditText android:id="@+id/edit_message"
          android:layout_weight="1"
          android:layout_width="0dp"
          android:layout_height="wrap_content"
          android:hint="@string/edit_message" />
     <Button
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
</LinearLayout>
```

The printWriter is declared in the top of the file, and is initialized as above. A printWriter is a writer that sends print formatted representations of objects to a text-output stream. This printwriter is sending the messages to the desktop-program which is described in section 4.2.1.

When implementing applications that are going to send information over the internet, a few important lines are important to add to the AndroidManifest.xml. This can be seen in listing 4.3, and allow the application to send data over the internet.

Listing 4.3: Permissions for internet

```
<uses-permission android:name="android.permission.
    INTERNET" >
    </uses-permission>
    <uses-permission android:name="android.permission.
        ACCESS_NETWORK_STATE" >
    </uses-permission>
```

The following items were implemented in this section:

- Runnable Thread, with a run function which runs continuously
  - class ClientThread implements Runnable
    - public void run()
- A function being called when send-button is pressed
  - public void onClick(View view)

4.2.3 Adding exception on port

This system was implemented on a public network, and with a firewall. This caused some problems when receiving data, so there was a need to add an exception on one of the ports. In Windows 7 this could be done with adding a rule in the firewall. To be able to do this, one has to run as administrator on the computer.

In order to add the exception one have to begin with opening the start menu and do as following according to Windows them selves[48]:

1. Choose **Control Panel** -> **System Security** -> **Windows firewall**
2. In the navigation pane, choose **Advanced Settings**

3. Choose **Inbound rules**, and then in the Actions pane choose **New Rule**
4. On the rule page choose **Port**, and then choose **Next**-button
5. On the **Protocol and Ports** page, select **Specific local ports**, and then enter the port number you want to add the exception to
6. Choose the **Next** button
7. On the **Action** page, choose the **Allow the connection**, and then select the **Next** button.
8. On the **Profile** page, choose the profiles, and then choose the **Next** button.
9. On the **Name page**, type a name for the rule, and then choose the **Finish** button

4.3 Displaying Messages

After the TCP-connection between the desktop program and the application was working the next challenge was to display the messages on each side.

4.3.1 Desktop Program

From section 4.2.1 the code that received and sent the text messages was implemented. So in this part the User Interface was added. This part of the program was not a part of the program, and therefore this User Interface became less important, and based largely on the suggestion on the web-page from Android Solutions[45].

The first thing done was to add a new class file to the Java project. The way to do this is explained in section 4.2.1. This file was named "MainScreen.java", and contained all the information about the user interface for the desktop program. The constructor of this class became *public MainScreen()*, which takes no arguments.

This constructor inflates the different parts of the screen. The Screen consists of a main frame that is the background for the items added onto it. As this was going to be a chat-program the main features that was needed was an area to enter messages, a area to display messages and a send-button.

To be able to do this the screen was split up into two parts as seen in figure 4.3. The most normal way to implement a user interface within a chat-program is to have the display of the messages on the top, and the area to type the message you want to send in the bottom. This is to mimic the way one would write on paper; one begin at the top and work your way downwards.

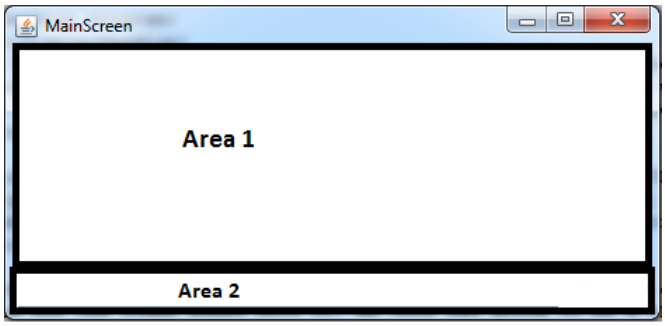


Figure 4.3: MainScreens areas

As Java has several pre finished packages to use, the JFrame package was a simple way of using the already predefined functions. In this was one could implement two panels as easy as seen in listing 4.4.

Listing 4.4: Implementing panels in Java

```
JPanel panelFields = new JPanel();
panelFields.setLayout(new BoxLayout(panelFields,
    BoxLayout.X_AXIS));

JPanel panelFields2 = new JPanel();
panelFields2.setLayout(new BoxLayout(panelFields2,
    BoxLayout.X_AXIS));
```

These panel fields could then be filled with the wanted features. In Area 1 the area to display the messages was inserted, and Area 2 got the area where one can write the message as well as the send-button, as seen in figure 4.4. The area that takes input is a simple JTextField, and the send-button is a JButton. The area that displays the messages is a JTextArea. All three of them are a part of the JFrame package. To be able to display this frame it needs to be called from the TCP-file which include the main()-function.

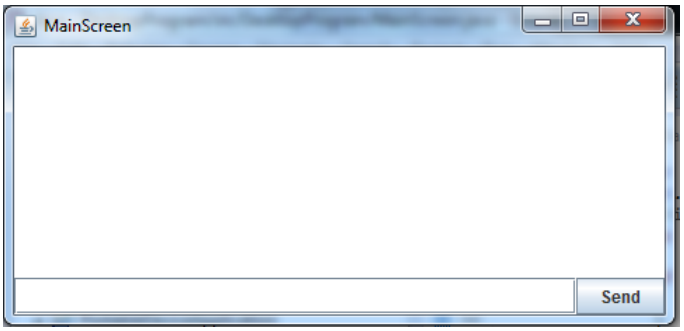


Figure 4.4: Main Screen with display area and input area

Another handy feature that was added to the Screen was the ability to start and stop the transferring of messages. The finished screen can be seen in figure 4.5. When pressing the start-button the TCP-connection is established and the listening for a receiving unit begins. When pressing the stop-button, the socket is closed and the transferring ends.

This makes the system sketch as in figure 4.6.

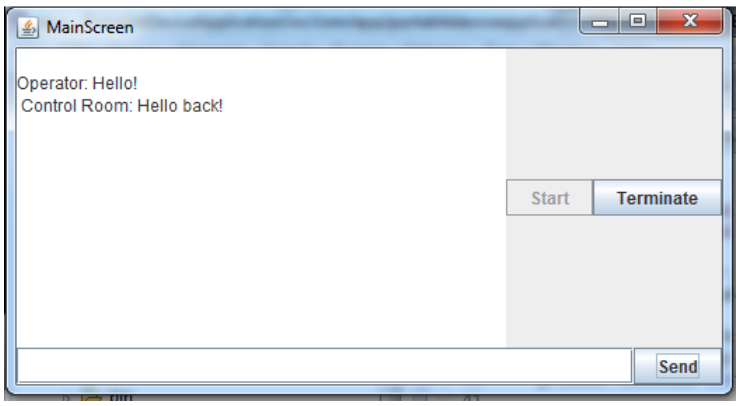


Figure 4.5: Finished Desktop User Interface

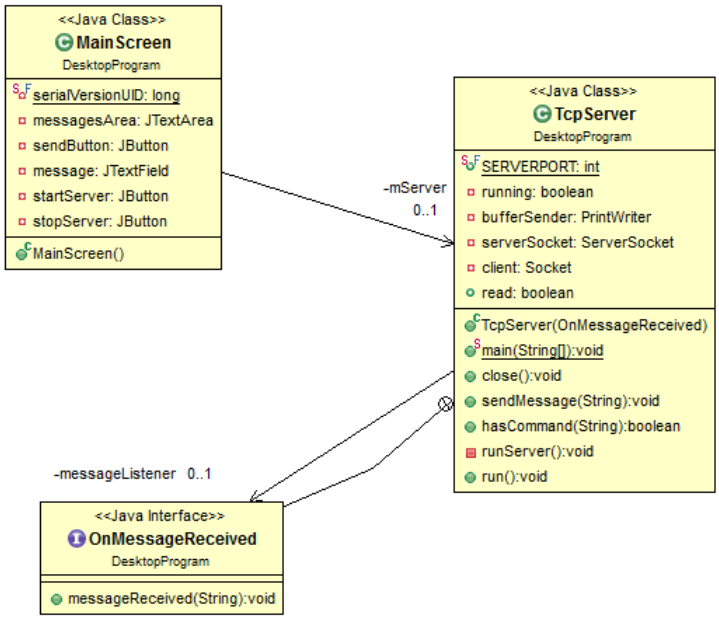


Figure 4.6: UML for Desktop Program

4.3.2 Portable Device

From section 4.2.2 the code to add a send-button as well as a thread that constantly listens for messages from the server. After this a way to show the different messages in the current activity needed to be implemented.

An easy way to show a number of text messages was to make a listView. This is an interface item that shows items in a vertically list, with the possibility of scrolling. To do so an *arrayList<String>* was chosen as the list-type. Code was added to both the client thread, as well as in the function triggered when the send button was pressed.

When pressing the send-button, the code will check if the EditText-field is empty. If it's not empty the text-input is added to the arrayList. The operation when a message is received from the desktop program is the same. The name of the origin of the message was added in front of the message, for example "Operator:" in front of the messages from the program them selves, and "Control Room" in front of the message. This was done statically using a StringBuilder and adding the strings to each other.

To be able to add s string to an arraylist one need an adapter. The adapter is

made as in code block 4.5.

Listing 4.5: ArrayAdapter Code

```
ArrayAdapter<String> arrayAdapter = new ArrayAdapter<
    String>(this,android.R.layout.simple_list_item_1,
    your_array_list);
```

This adapter use the standard arrayAdapter included into the arrayList-library. First one needs to add a listView to the xml-file displaying the layout of the current activity. Thereafter one needed to link that listView to the correct arrayList in the Java code. This can be seen in code block 4.6.

Listing 4.6: Connecting ListView and ArrayList

```
ArrayList<String> your_array_list = new ArrayList<String
    >();
private ListView lv;
lv = (ListView) findViewById(R.id.your_list_view_id);
your_array_list.add(message);
ArrayAdapter<String> arrayAdapter = new ArrayAdapter<
    String>(this,android.R.layout.simple_list_item_1,
    your_array_list);
lv.setAdapter(arrayAdapter);
```

The user interface of this implementation the can be seen in figure 4.7.

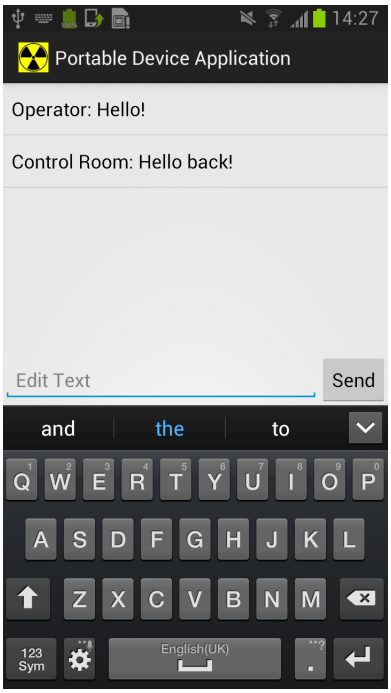


Figure 4.7: Simple ListView

When this was added the system sketch was as shown in figure 4.8.

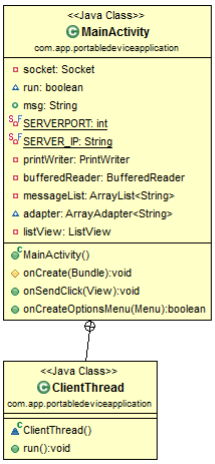


Figure 4.8: UML for Portable Unit Application

4.4 Expanding User Interface for Chat

When the simple display of the messages as well as the correct transfer of messages was implemented, a more advanced user interface needed to be implemented to improve the user experience. One of the problems with a basic arrayAdapter, is that it is only possible to inflate a listView with one string input. To display more information in one list row, one would need several information segments.

As presented in chapter 3 the needed information for each chat-item is when the message arrived and the message it self. In addition the name of the sender was added. All these three items can be resolved to strings, and it was therefore added two new arrayLists to the code. This was to be able to store the time of the delivery or sending of the messages as well as the sender-id.

In order to inflate more than one list item value for each row, a custom arrayAdapter needed to be implemented. To do so, one needs to add a new file. This is done by right click the project package, **Add New** -> **Class**. In this case a new class need to be implemented. This file needs a constructor, in which the inputs are turned into local variables. The input to the adapter is the different arrayLists. The id for the corresponding row xml-file, and the context of the main activity should also be inputs to the custom arrayList adapter. The class also need a *public View getView(int position, View convertView, ViewGroup parent)* and a xml file with the layout for each listView row.

To make the scrolling of the list smoother, a ViewHolder was added to the code. This holder stores each of the component views inside the tag field of the layout, so that it is possible to access them immediately without looking them up all the time.

This custom arrayAdapter will be called every time someone touch the screen. To trigger the update of layout is done by calling the function *notifyDatasetChanged()*, which will run the getView-function one time for each row in the arrayLists.

In this part of the application, the background of the layout was made as well. This layout was made to resemble the layout from chapter 3. Therefore the background was set to so called speech-bubbles. The finished layout can be seen in figure 4.9 and the updated class diagram can be seen in figure 4.10.

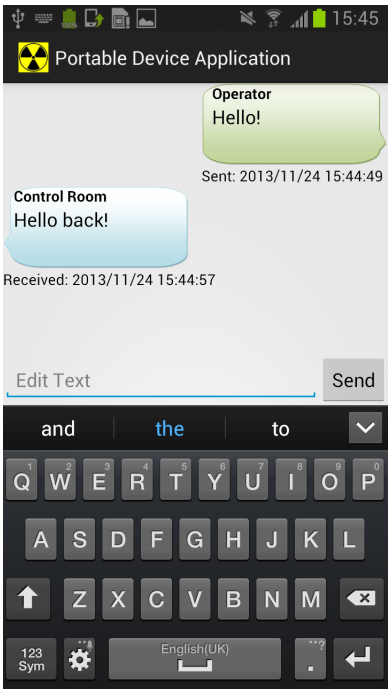


Figure 4.9: Screen shot with expanded user interface

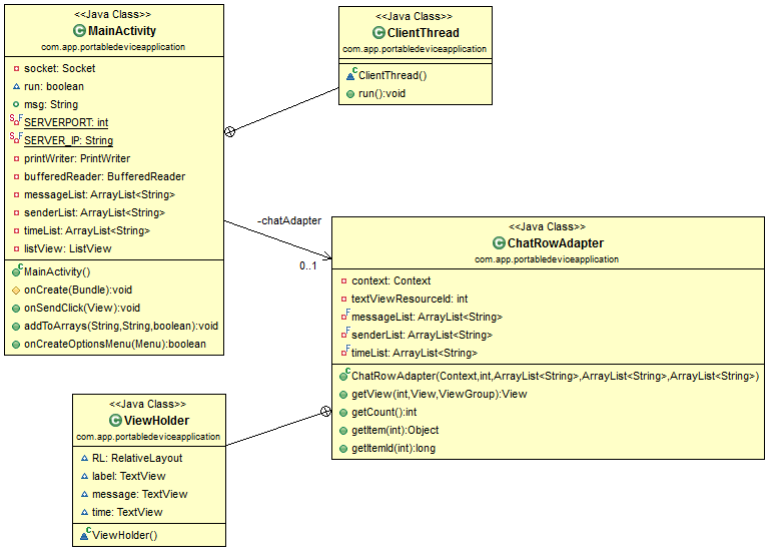


Figure 4.10: Class diagram for application with new interface

4.5 Stored Categories

To avoid confusion it was suggested in chapter 3 that some stored phrases should be made, which could make the quality of the conversations better.

The best way to do this is to make a new activity. When working in Eclipse one could easily make a new Android activity by adding it from the menu. This is done in the same manner as when adding a new class, except in this case one press "Other" in stead of "Class", and then choose Android Activity in the Android folder. This creates a new Java file, as well as a xml file in the layout folder.

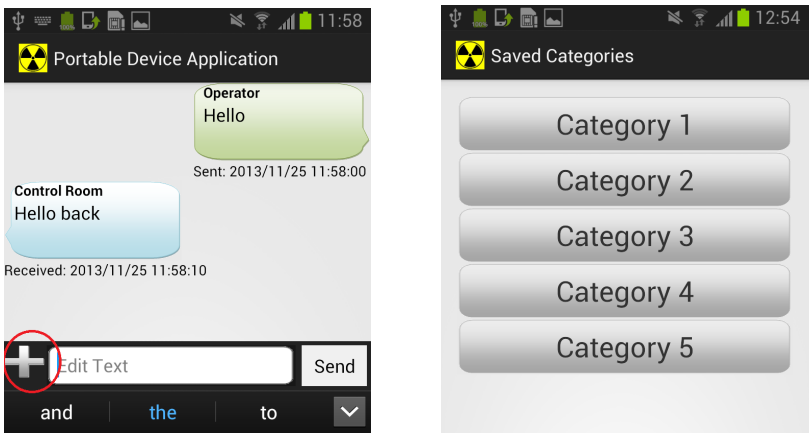
In order to start the activity an intent is made. An intent is an abstract definition of an operation to be performed[49]. It provides a facility for performing late runtime binding between the code in the different applications. There is two ways of starting an activity; one is to start the activity normally, and the other is to start an activity for a result. These are displayed in listing 4.7.

Listing 4.7: Start an Activity for results

```
Intent intent = new Intent(context, PreStoredCategories.
    class);
startActivity(intent);
OR
startActivityForResult(intent,int requestCode);
```

The wanted result in this section is a string with the selected phrase. To collect these result an *onActivityResult(int requestCode, int resultCode, Intent data)*-function need to be implemented. To prevent the application of crashing it is necessary to make a switch on the resultCode. The result could be returned as RESULT_OK or RESULT_CANCELED. The latter option will occur if the activity is aborted before a phrase is selected, making the returning data *null*. This happens if the user for example press the "go back"-button. To trigger the start of the new activity the plus-button in figure 4.11a was added. When this button in pressed a method is called and the new activity is instantiated.

This layout needs a list of categories to be displayed in a simple listView. This is done as a standard arrayAdapter, as seen in section 4.3 and displayed in figure 4.11b. The changes made in this section led to some new functions, and the class diagram of the affected files can be seen in figure 4.12.



(a) Saved Phrases Button (b) Category List

Figure 4.11: Category List Screen Shot

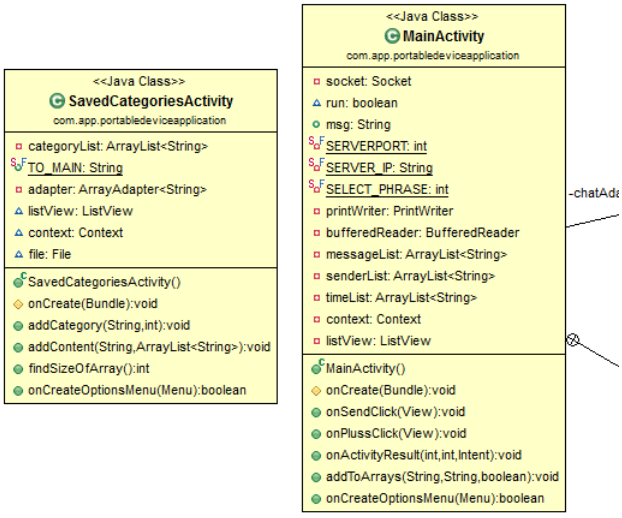


Figure 4.12: UML for updated files in this section

4.6 Stored Category Content

When the stored categories was implemented in section 4.5 there was a need to fill these categories, and to display the content in a similar layout as the one

displaying the category-list them selves.

In order of avoiding one activity per category, one needed to hold all of the data in the Stored Category class, and thereafter pass the selected category content to the new activity according to the category chosen. To hold the arrayLists that needed to be sent to the new activity an *ArrayList<ArrayList<String»* was needed. For listViews it is possible to add an item click listener, which returns the position of the item selected. This can be seen in code section 4.8.

Listing 4.8: Listen for pressed item in ListView

```
listView.setOnItemClickListener(new OnItemClickListener()
    {
@Override
public void onItemClick(AdapterView<?> parent, View view,
    int position, long id){
  }
});
```

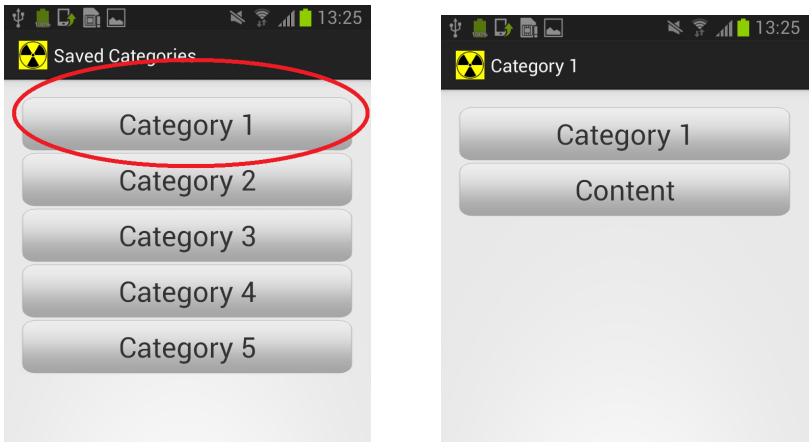
Subsequently when an item in the arraylist displayed in the "Stored Category Activity" was pressed, the content in the arraylist that holds the other arraylists for the same position was sent to the "Stored Category Content Activity". To send data to another activity, one can send it along with the intent that starts the new activity. This is done simply by the *intent.putExtra()*. The data that are going to be sent need to be accessed through an address such as the url for you application, *"com.example.myApp.NEW_ARRAY"* stored in a string. This string is then sent along with the intent, so that one can access the data from the new activity.

In the Stored Category Content Activity the data needed to be received. This is done with the code seen in code section 4.9.

Listing 4.9: Receive data from another activity

```
Intent intent = getIntent();
ArrayList<String> contentList=intent.
    getStringArrayListExtra(PreStoredCategories.NEW_ARRAY)
    ;
```

When the data is received the list is displayed with the standard listView, and arrayAdapter. This finished result is shown in 4.13.



(a) Pressing item in list (b) Category Content List

Figure 4.13: Category Content List Screen Shot

When the layout was in place, a item click listener was implemented for this listView as well. This item listener would trigger the listener, and send the chosen string via the "Stored Category"-activity, back to the Main Activity. To do so is as seen in the following code block 4.10. This code also was added to the onActivityResult-function in the Stored Category Activity.

Listing 4.10: Activity Finished

```
String content = contentList.get(position);
Intent intent = new Intent(context, PreStoredCategories.
    class);
intent.putExtra(EXTRA_CONTENT, content);
setResult(RESULT_OK, intent);
finish();
```

When returning to the Main Activity the selected data was inserted into the EditText that display the message to be sent. In case the phrase does not fit completely, there was a possibility to alter it.

The updated class diagram for the affected files can be seen in figure 4.14.

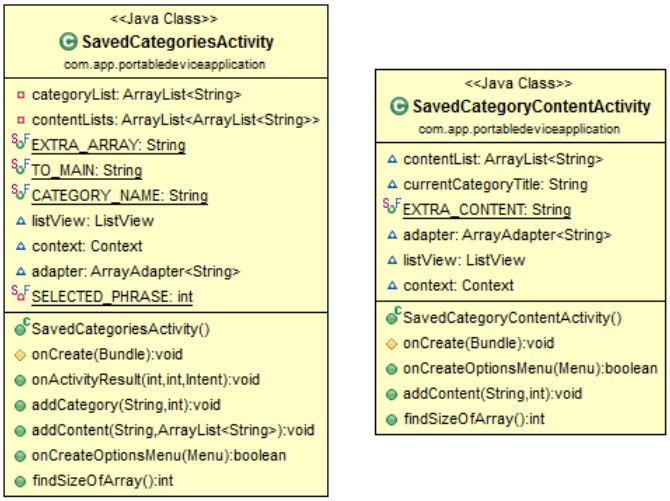


Figure 4.14: UML Class Diagram for affected classes

4.7 Read Confirmation

As described in chapter 3 a read confirmation was needed to be added to the program. This would trigger changes in both the desktop program and the device application, explained in this section.

4.7.1 Desktop Program

To be able to see when the supervisors in the control room had seen the message there was several possibilities to detect when the supervisors interfered with the user interface on the desktop. One could implement a focus-listener, a mouse-listener, or a window-listener. The difference between them can be seen in table 4.4.

The listener that was chosen was the mouse listener. This listener was implemented as an extension to the MainScreen Class. The functions needed to be implemented in order to make the listener work is listed in code section 4.11. If these are not implemented the compiler will complain.

Listing 4.11: Mouse Listener

```
public void mouseClicked(MouseEvent event){
}
public void mouseEntered(MouseEvent event){
```

```
}
public void mousePressed(MouseEvent event){
}
public void mouseExited(MouseEvent event){
}
public void mouseReleased(MouseEvent event){
}
```

When these functions was implemented the listener was set on the frame containing the messages. To notify that the supervisor in the control room have read the message, the supervisor only need enter the frame with the mouse pointer over the area where the messages is displayed. One of the arguments from chapter 3 was that the read arguments would simulate a nod, or a similar way of responding that the message was received. To make this response one need to actively do it, and in a similar manner the control room operators need to actively move the mouse pointer into the window, and it does not happen automatically.

To do so the mouse pointer enters the message area and the mouseEntered-function is called. In this function one sends the read confirmation to the portable unit. One problem with this is that the mouseEntered() function will be called every time the mouse pointer enters the frame. This can be more often than the frequency of new messages, and can cause the connection to the portable unit being flooded. In order of avoiding this a *boolean* variable was added, that indicated that the read-message will only be sent if there is messages that are unread.

The class diagram with the files updated in this section is displayed in section 4.15.

Focus listener	Focus events are triggered when a unit gains or loses the keyboard focus. This is true whether the change of focus happen with the mouse, the keyboard or programmatically.
Mouse Listener	Mouse events are triggered when the user uses the mouse to interact with a component. The activities that invoke an event are: • The mouse enter the item. • The mouse leaves the item. • The mouse presses an item. • The mouse release the item it as released. • The mouse is clicked.
Window listener	This listener listens for window events. These are invoked by the actions: • The window is set to be the active window. • The window has been closed. • The user has pressed the X in the corner to close the window. • The window is no longer the active one. • When the window is changed from minimized to normal size. • When the window is changed from normal to minimized size. • The first time the window is available.

Table 4.4: Difference between listeners

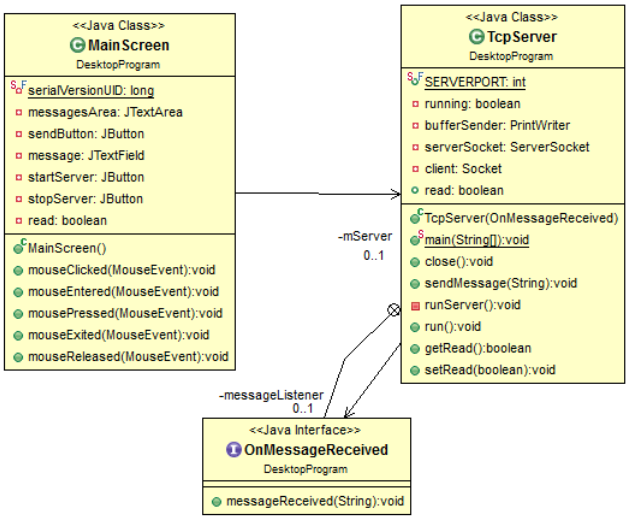


Figure 4.15: Class Diagram for Desktop Program

4.7.2 Portable Device

After the read confirmation was sent from the desktop program it needed to be received and handled in the application on the portable device. The variable it self will be received in the same place as the other messages from the desktop program.

To display that the message was read, a check-mark was added beside the date and time it was sent. These pictures reserved some space, and to keep the user interface as clean as possible the check-marks is only added when the message is the one sent from ones own unit. In addition the time-string was changed from "Sent yyyy/mm/dd hh:mm:ss" to "Read yyyy/mm/dd hh:mm:ss". In order of keeping track of which messages that were read and which that were not read a new ArrayList<Boolean> was added. This array hold boolean values of which messages is read. When the read-update from the desktop arrives all messages with false values in this boolean arraylist will have their time string updated.

As the desktop version of the program was not a part of this assignment there was no read-notification sent from the portable device. However it is fairly simple to implement a focus listener on each field. This is done a similar manner as the onItemClickListener from section 4.5, except it need to be an setOnFocusChangeListener, and the condition *hasFocus* need to be fulfilled.

When this was finished the user interface of the application looked as shown in figure 4.16, and the class diagram as in figure 4.17.

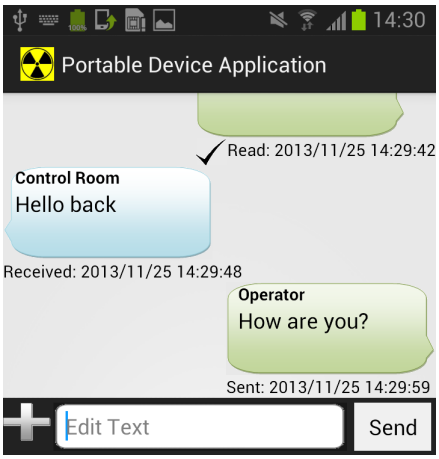


Figure 4.16: Chat User Interface with read-confirmation

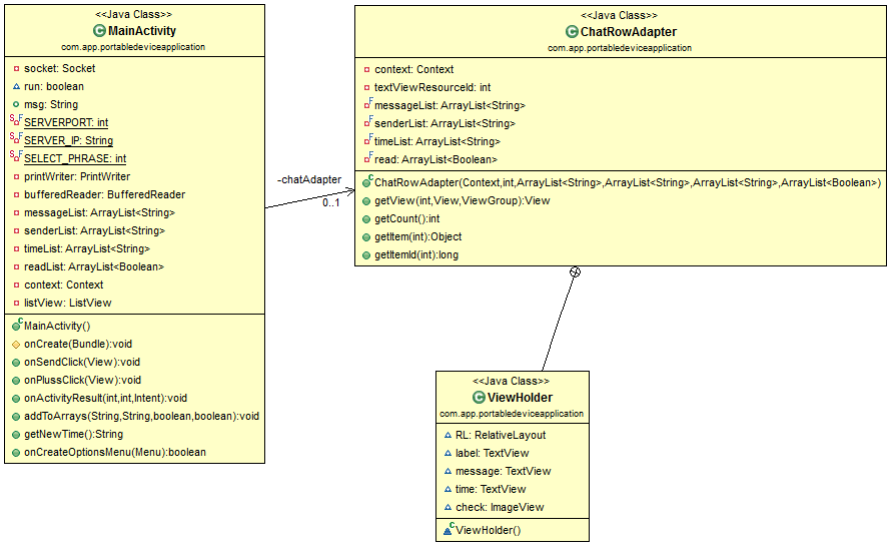


Figure 4.17: Class Diagram for Application after read check

4.8 Sorting of Messages by ID

After some time with conversation on this application the messages within the focus-area will be severely limited compared to the total amount of messages. To be able to look for a specific message one will spend some time looking for

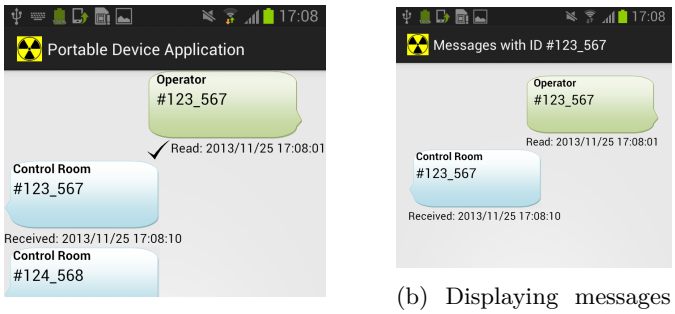
it. To reduce this time it was suggested in chapter 3 that a sorting of messages regarding each unit should be possible.

In a NPP each unit will have an identification number. This number is built of a section with three numbers, then follows a space, then two letters and last a undefined number of numbers. The fist number is determined by which system the unit is a part of. Per example the turbine-units will start with a four, and the reactor units will start with a two.

One symbol that is rarely used in most languages is #. This symbol is used in several social networks as a prefix when creating a link to a subject. When using social networks one can then press this link and see all other messages regarding the selected subject. As this is mainly what was needed to happen in this application as well, the #-symbol seemed as a suitable prefix for the identification numbers.

To choose a specific identification number, one needs to press a chat-bubble that contains the relevant id-number. The way to implement a *onItemClickListener* was shown in section 4.5, and when this function was triggered the program ran through the message of the specific position. If the message contained a #-symbol, a loop would run through the message and withdraw the characters following that symbol. When a space occurred the phrase would be stored in a new string, and this would be the id. This means that the three first numbers and the rest of the id needed to be separated by an underscore in stead of a space.

When the id was found, a new for-loop would search through all the messages. With the handy command *string.contain(string)* one can get a "true" every time the message contained the id-numbers in the initially selected message. Each time the value "true" was returned a new method would be called and the content of all original arrayLists for the specific position was added to new arrayLists. These arrayLists were therafter sent to a new activity. This activity is called DisplayMessagesWithID, and need to have the same layout and row adapter as the MainActivity. This activity does not need to be started with the intent of wanting a result as the activity is a result in it self. The new activity can be seen in figure 4.18. Figure 4.18a display the original chat with all messages, and figure 4.18b display only messages with the selected id.



(a) Original chat layout

(b) Displaying messages with specific ID

Figure 4.18: Before and after pressing message for ID-sorting

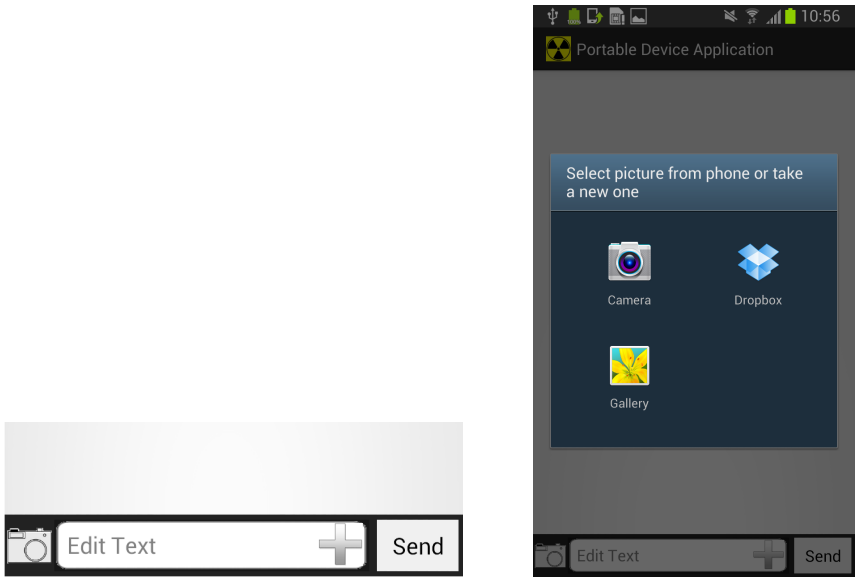
Some times a message would contain no sort of identification, and therefore nothing would happen when pressing it. However the feature was added so that only the message it self would be displayed when pressing it, though in bigger font.

At this point the UML-sketches were too large to be displayed for each update. The UML for the total system can be seen in section 4.13.

4.9 Adding Pictures to Chat

"Use a picture. It's worth a thousand words", as stated by newspaper editor Arthur Brisbane[50], would sometimes be an overstatement. But that a person would quickly absorb large amount of information through a picture is certainly true. Therefore it would be an advantage to add the possibility to send pictures from the field operator to the supervisors situated in the control room.

To trigger this event a new button need to be added. In almost every application in section 2.5 a camera-picture is set as background of such a button. Thus this was added in this application as well. When the button is pushed a function is triggered. This function is then starting a intent for a result, and with a result code. However this time a new activity is not needed. When pressing the button as seen in figure 4.19a, the intent started is as shown in 4.19b. The buttons have been moved around from 4.5. The plus-sign is however no longer a button but a clickable area of the EditText-field.



(a) Picture button (b) Picture intent

Figure 4.19: Adding a new picture

When selecting one of the programs shown in figure 4.19b one would be able to either select a stored picture or take a new one. Dropbox is as seen in appendix B an included place to store files in the sky. The file-path Uri of the selected picture is then stored in the intent when returning to MainActivity in the *onActivityResult()*. In order to obtain the picture it self one need to look up the picture in the system. This would return a bitmap that can be displayed. To show the picture properly the orientation is also needed. This is so that the picture would not bee shown vertically if the portable device was held horizontally when the picture was taken.

To exhibit the picture in the chat application an arrayList<Bitmap> was added to the already existing code. A new ImageView was also added to the xml file that contains the layout of each row of the chat application. In order of being sure that all these arrayLists kept the same sizes and that the content was correctly matched with the others was to implement a addToAllArrays()-function. This function would add one value to each array as it was called. Code to deal with the pictures was also added to the sorting feature, displayed in section 4.8.

The new bitmap retrieved from the intent was then stored in its own array and displayed in the chat-application as shown in figure 4.20. At this point some padding was added to the speech-bubbles to avoid that the content would be displayed outside the bubble.

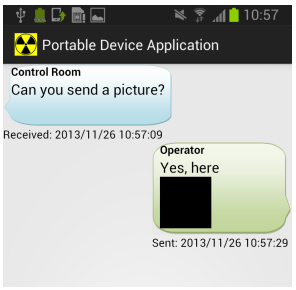


Figure 4.20: Chat with pictures

What was soon discovered when adding these pictures was that pictures take a lot of space in memory for the application. For the small images needed to be displayed in this chat-application the full picture was not needed to be displayed. Therefore a thumbnail of the picture was needed to be displayed in the "Main Activity". The thumbnail in this example is a simple black picture that is meant as an illustration.

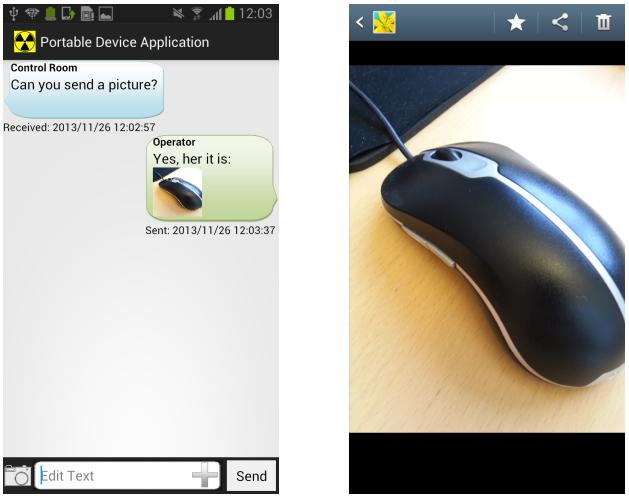
It is however not possible to see any details in a picture of such a small size and the choice to have a function that display the picture on a full-screen was made. As it exists a *setOnItemClickListener()*-function it also exist a *setOnItemLongClickListener()*. This function will then trigger the display of the picture in the Picture Gallery that is in the phone. To be able to obtain the picture it-selves the uri was saved in an arraylist in the same manner as the other arrayslists. The way to do this is displayed in code section 4.12.

Listing 4.12: setOnItemLongClickListener()

```
listView.setOnItemLongClickListener(new
    OnItemLongClickListener(){
        @Override
        public boolean onItemLongClick(AdapterView<?>
            parent, View view, int position, long id) {
          Intent intent = new Intent();
          intent.setAction(Intent.ACTION_VIEW);
          intent.setDataAndType(uriList.get(position)=, "
              image/*");
          intent.putExtra(messageList.get(position),
              pictureList.get(position));
          startActivity(intent);
          return true;
        }
      });
```

The displayed picture will look as seen in figure 4.21. Figure 4.21a display the chat view with messages included with pictures. By holding the second chat

item, the view in figure 4.21b will appear.



(a) Chat with pictures (b) DisplayPicture

Figure 4.21: From chat to picture display

4.10 Adding Categories or Phrases

From chapter 3 one of the improvements was to include the saved phrases. Although some of the content in the stored categories was planned to be finished in advance the opportunity of adding content was also added.

A way to make this possible is to add an option of "Add new content" in the end of every category, and in the category list make an "Add new category". As all of the content of the categories was handled in the "Stored Category"-activity this was were most of the changes was implemented.

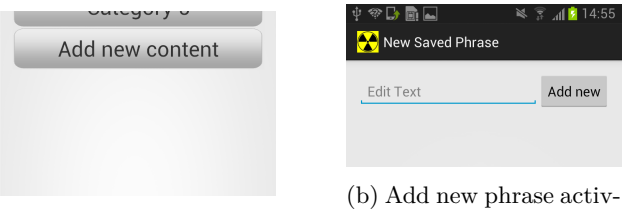
In Android layout there is a ViewSwitcher available. At first it seemed as a good idea to add a ViewSwitcher to the last item in the array, so that one could switch between an EditText-layout and a standard TextView. This item would however be added to all of the items in the list. To only activate this for the last item, an *onItemLongClickListener* was added, and which only activated the switch for the last item.

However the ViewSwitcher will be called every time the getView in the adapter is called. And after some testing and failing, the result was that the ViewSwitcher altered the behaviour of the application so that there was changed views where it should not be, and as the normal *onItemClickListener* would react when trying to insert text. Therefore the ViewSwitcher was deemed as not an appropriated choice for the use of this feature. A new activity was then implemented only

with the layout shown in section 4.2.2. This new activity also needed to be called from the Saved Category Content.

This activity would return a string. When pressing the "Add new ..."-button in the category list a new category would be added to the list of categories, and a new arraylist would be added to the list of arraylists, along with the entry "Add new content". When adding new content, the content was added to the arraylist in question. To do so the new string was sent back to the Saved Category Activity and added to the correct arrayList there.

The new button are shown in figure 4.22a, with the triggered activity in figure 4.22b.



(a) Add new phrase

(b) Add new phrase activity

Figure 4.22: Adding new category or content

4.11 Saving Phrases Locally

Android activities are managed as an activity stack. When a new activity is started it is placed on the top of this stack, and will remain so until another is started, or the activity it self exits[10]. The life cycle of an activity is shown in figure 4.23.

When entering the "Stored Category"-activity the first time, all the entries are statically coded into the *onCreate* method, and added to the list with static strings. With the previous section 4.10 a problem arises. If one have added a new category or new category content this exists when alters between these two activities, but when entering it later from the Main Activity it is lost.

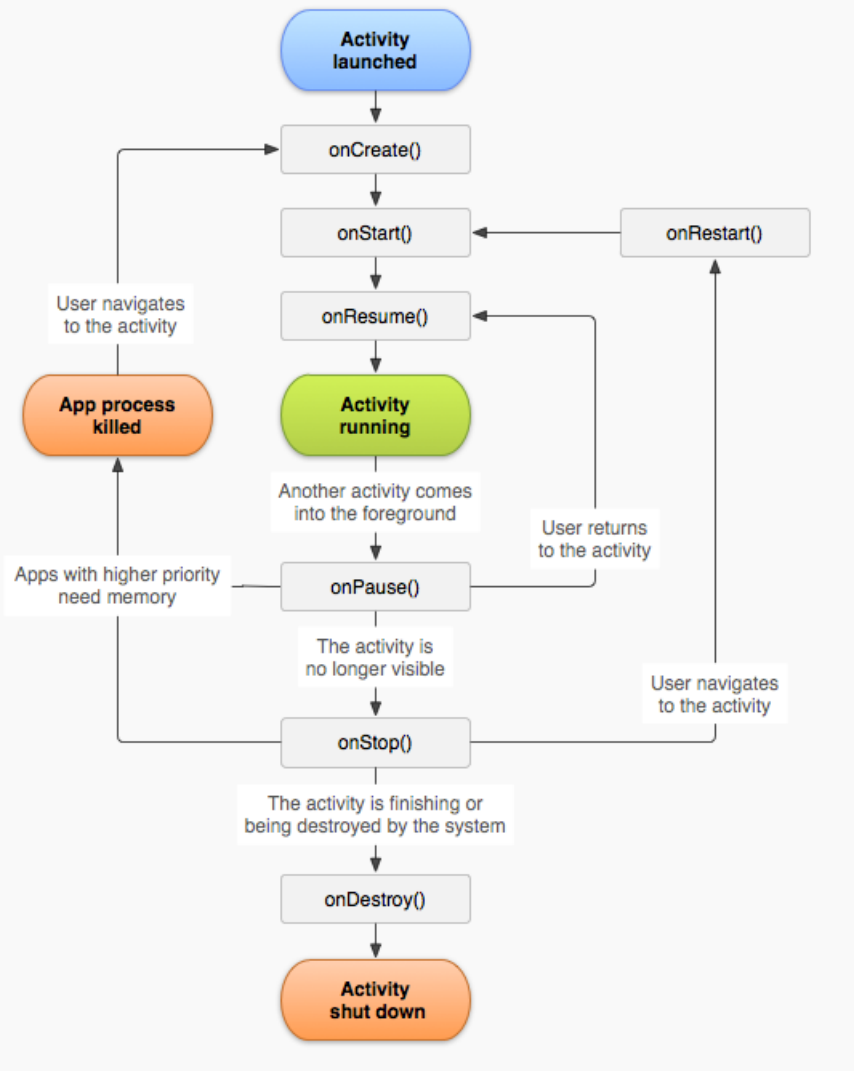


Figure 4.23: Android Activity Life Cycle [10]

As the intention is to have a set of previously made statements in the application as well as the possibility of adding new categories or content, this needs to be changed. The solution is to remove the static strings from the program and in stead make a resource folder with the saved phrases. These phrases was stored in a .txt-file and with the interpretation as shown in table 4.5. The reason the first item has a 0 and not a 1 in front of it is that the arrayLists is zero-indexed. The underscore works as an separator, and one can therefore have categories or content with several words.

Input from text file	Meaning
0_Category1	A new category, to be placed first in the list
0_0_Category1Content1	New content, to be placed first in the first category
1_Category2	New category to be placed second in the category list

Table 4.5: Style of text input

When starting the program for the first time, the content of the phrases is to be read from this .txt file. But it is not possible to write to this file in runtime. Therefore a local file needed to be established, so that new content or categories could be saved in this file. The first time the activity was run the content of the .txt file was read and then written to this local file. After this it was necessary to check which file should be read from. In order to do so the sizes of each files was compared and the biggest file was read from when populating the arrays. Every time an item was added it was also written to the local file.

These methods was placed in its own class, and accessed with a object that access the available methods.

To be able to write to the local storage the code in listing 4.13 needed to be added to the Android Manifest. This would grant access to the external storage in the SD-card.

Listing 4.13: PermissionStorage

```
<uses-permission android:name="android.permission.
    WRITE_EXTERNAL_STORAGE" />
```

4.12 Saving Messages Locally

The same problem that occurred in section 4.11 occurred for the Main Activity when the entire application was started over again. The code for storing the saved phrases and the messages was pretty much alike, except the different file names. Therefore most of the code used to save the phrases and categories could be used to save the messages, and the other new methods were added to the same class.

The first time the application is run, no messages are stored, and so it is not necessary to read from the text-file. Code was added to check if the message file in the local SD-card was empty and if it was not empty, messages was read from it.

Messages was added to the text file in the same manner as with the stored phrases seen in section 4.11. Each line was one message entry, with the content of all of the arrays. The decoding logic of this item was placed in a separate method in the SaveLocally-class.

All of the content of each row item was added together with a StringBuilder that added an underscore between each of the items. The messages were added like this: message_sender_Time String_Bolean if it was read or not_File-path. The picture was not added, and therefore replaced by the corresponding file path. This file path needed to be checked if it was there or not, and with no picture present *null* was added.

This was the last feature added to the application and the class diagram become as in section 4.13.

4.13 System sketch, UML

The total system sketch for the portable unit is seen in figures 4.25 and 4.26, and the full class digram for the Desktop Program can be seen in 4.24. This system is not completely similar to the suggestions in chapter 3, and the differences is discussed in chapter 6. To obtain this code, one can use the CD included in this Master's Thesis, and follow the instructions from appendix C.

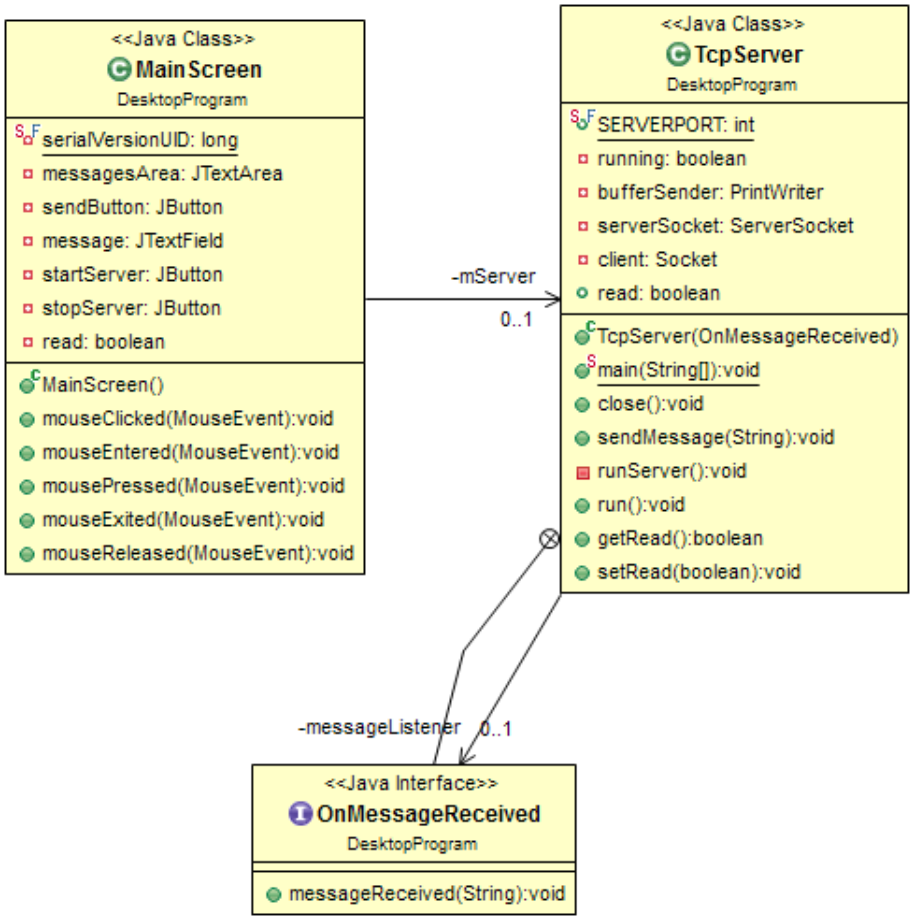


Figure 4.24: Full UML for Desktop Program

Chapter 5

Content of pre-stored phrases

5.1 Phrases from papers

As it was suggested in chapter 3, saved phrases was implemented in chapter 4. When this was done there was a need to figure out what these categories should be and the following content of them. Searches at the internet did display some examples of communication between a field operator and a supervisor in the control room, but there were not enough examples to find any finished sets of communication.

One of the articles explained in section 2.3 [2] explained a lot of the communication, but only had specific examples of the communication inside the control room. Otherwise there was a lot of articles on the communication during an emergency. That the operators will take the time to type a message in stead of calling seems unlikely.

When looking at the examples in appendix A the examples both circulate around units within one system. Therefore the logical solution would be that each category in the saved category list is one system inside the plant. Suggested categories will be :

- Boilers
- Steam Turbine Generators
- Chillers
- Water Plant
- Air Compressors

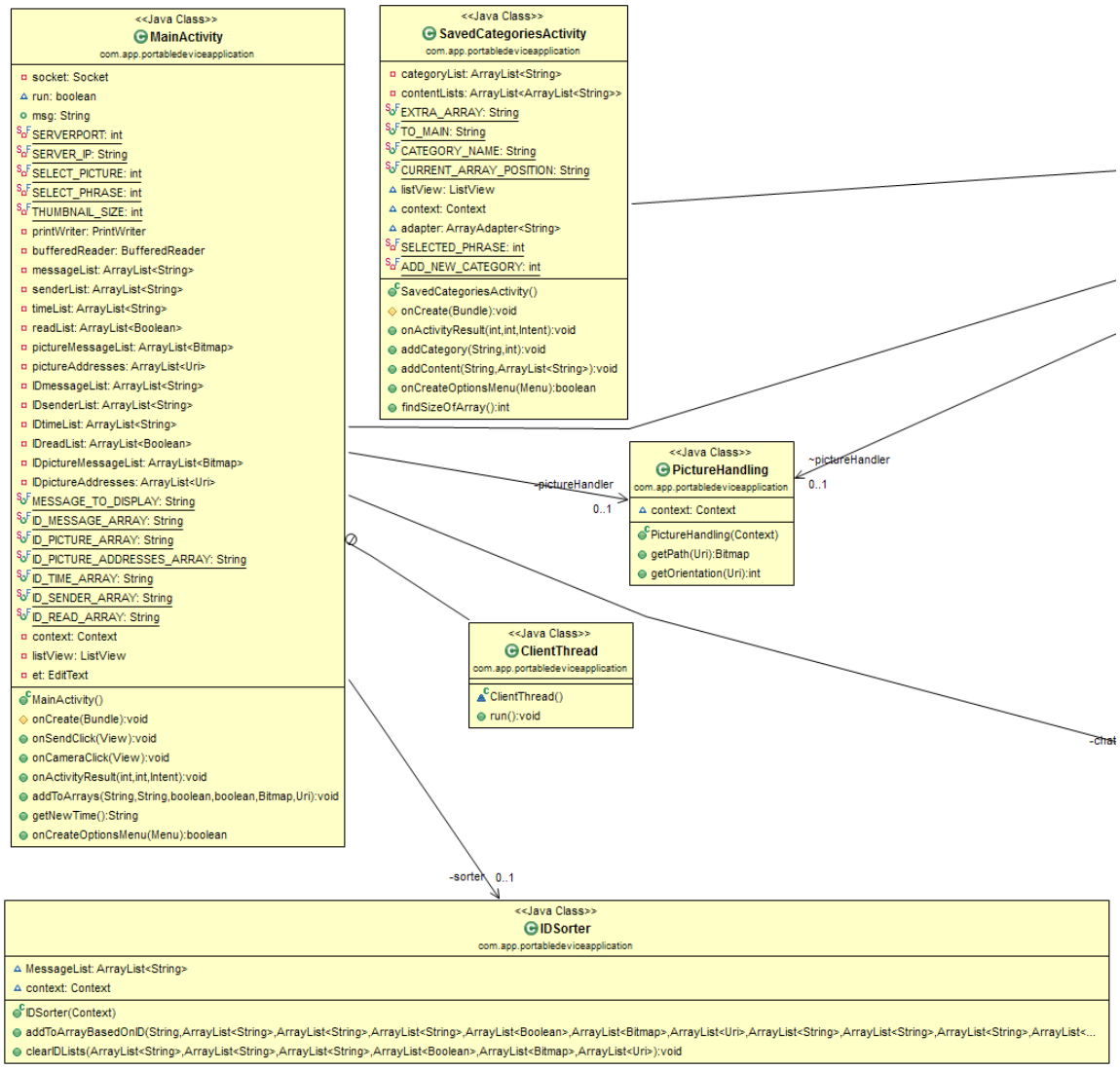


Figure 4.25: UML for Portable Application, left side

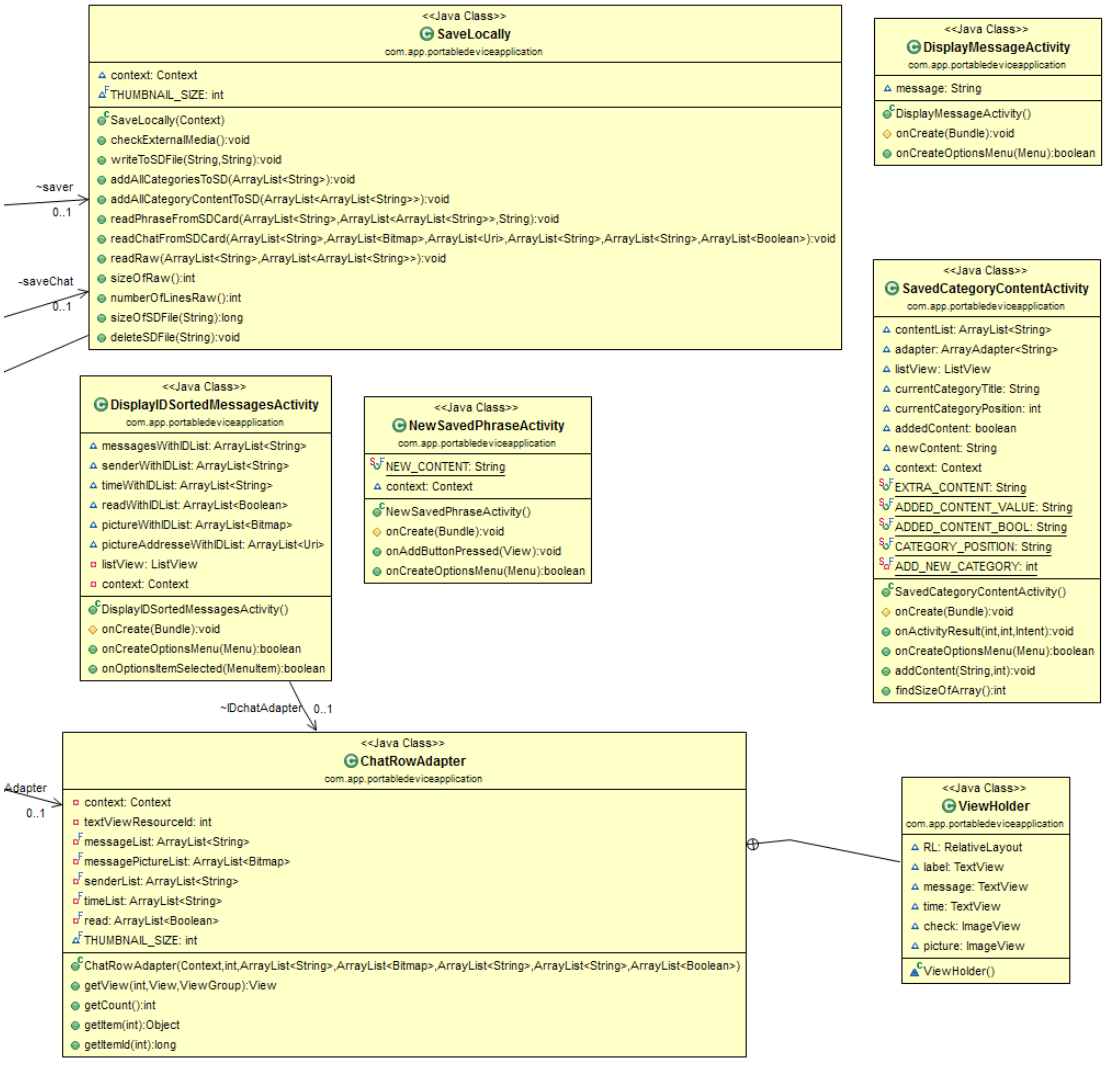


Figure 4.26: UML for Portable Application, right side

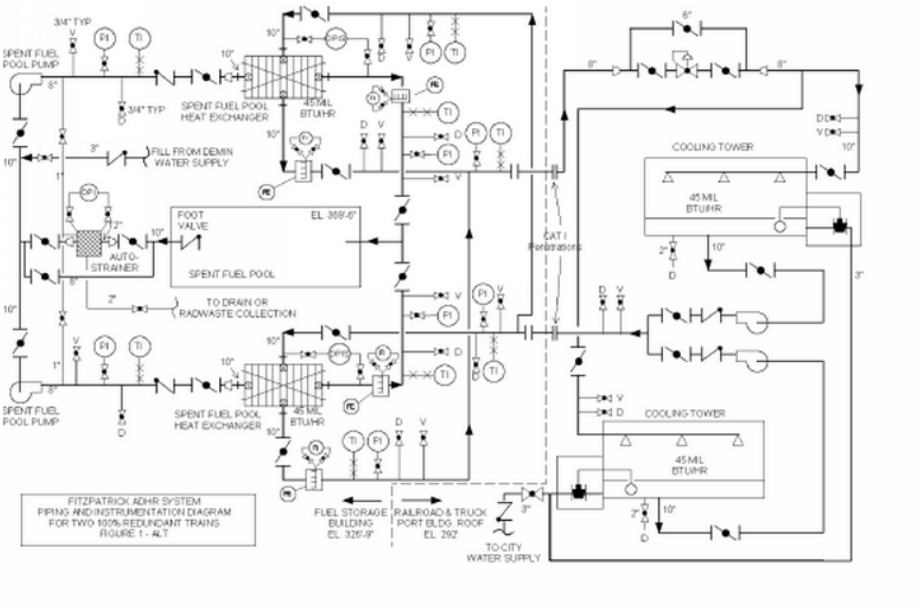


Figure 5.1: Piping and Instrument Diagram, example[11]

- Cooling Towers

These categories might be a bit unspecific, and one might need to split them into levels, as the plant would for example have more than one boiler.

When adding content to these categories one would have a look at the piping and instrumentation diagram for the specific system. An example of one of these diagrams is displayed in figure 5.1. This example displays only the cooling system in a power plant, and gives a sense of how many power plant units there actually is. To make a general suggestion for this system would be very difficult for someone that has not sat ones foot inside a power plant. To make these phrases it would be better to have someone that knows the system explain what phrases they use most commonly. As one have the option of adding categories and content it is also possible to add commonly used phrases to the categories.

In addition to the categories concerning the specific elemts it should be some categories with the more common phrases. These would be possible to edit in aftermath, and should contain phrases and categories like:

- Confirmation
  - I how now turned on...
  - I have now checked the level on...
- Refutation

  - I do not have time for that now, please contact someone else
  - I do not understand what you are asking me to do
- Time estimates
  - I think I will use about ... minutes
  - I will do it immediately

5.2 Interview with an operator

After the implementation was finished an interview with an operator was scheduled. This interview was held to get the operators view on the application, and which phrases he though was necessary. The interview was held 18th November 2013, between the author, Svein Nilsen(supervisor) and Håkan Svengren(operator). Håkan Svengren is a former control room operator and is currently employed by IFE, developing new power plant technology. After this he is known as "the operator".

The operator first explained how the communication between the field operator and the control room is carried out. This was mainly in the same manner as in section 2.3. In a NPP it is a number of procedures that it determined in advance. These are stored in a binder and kept at both the plant and in the control room. Usually the actions that are to be carried out during the day are determined in advance and the field operators are given a list of tasks. When the field operator receives a task, the most common acknowledgement that he or she has understood the order is to repeat it but to change it to "I am going to...". In this manner the operator felt that it could be useful to implement a copy-function that added the content of the received message to the edit text area, and one could from there edit it to be an answer. He was however worried that some of the messages would be rather long, and felt hat the longer the messages was the higher the risk of writing the wrong message became. As it is extremely important to send the correct message, the frequent mistyping would be a disadvantage.

When the field operator is employed it is no employment description that contained how the field operator should communicate, and no other written papers that tell them how to communicate. As a general manner each field operator is taught how to communicate and what each message mean by memorizing the content of the procedure binders located in the plant, and with help from control room and other operators. He went on explaining that there are no written instructions on communication inside the plant, except for when an alarm goes off. Then there is decided how one is supposed to communicate. He felt however that if there was an emergency one would rather pick up the phone than use valuable time typing a message. The reason for the lack of documentation and

regulations on communication was, in his perception, that it was impossible to write general rules for a system with ten-thousands of components.

The operator thereby confirmed the suspicions from section5.1 that it is not possible to limit the communication inside a plant to just a few phrases. He also feels that the possibility of watching messages regarding only one unit is not necessary, as the same message rarely occurs twice, and the number of ids is so massive. If these messages are a pickup for a more serious problem he feels that the control room will see it before one can see it in the messages. The control room will almost always have more information about the overall process than each field operator that see only small parts of the system.

Even though the operator felt the need to add tags to the message to see all messages regarding a certain unit was none-existent, he suggested that the pre-stored phrases should be switched with lists of ids. This list would be based on categories, where each category was one system (per example turbine units), and the content of each category would be the ids in this system. He felt that these tags should be making it possible to add the message as a task in the task list, and the id would link the message to the right item. The supervisor however was against adding to the existing task-list, but suggested that it should be a meta-list available only on the phone that had added the item.

Once opening up for saving the messages sent and received locally, one also open for the possibility of surveillance of each person. This was regarded as not applicable for this system by the operator. He explained that the control room can see everything that happens in the system anyway, and that each change in the system is logged. When saving the messages the field operators and the operators in the control room is given the possibility of proving what was said, if it comes to a discussion about it at a later point. It is however not often that such misunderstanding occurs. Most frequently the mistakes are faults that are easily detectable by the other part, and therefore corrected immediately.

One of the things the operator liked about the application was the ability to send multimedia. He felt however that it should be added the possibility of video-feed and pure sound files. The supervisor then suggested that one should include a sound-to-text converter, but the operator felt that this would only cause a lot of misunderstandings and misspelling. He was however enthusiastic about the possibility of sending sound-files, as seen in section 2.5.2. He based this on the fact that a sound file in addition to a read confirmation would simulate a conversation without having to relocate to the phone. Although this was considered a good idea, one major uncertainty in a power plant is the ration of noise. Testing is needed to see if the current noise-cancelling technology would remove the enormous amount of noise inside a power plant from the sound file. De Carvalho[24] stated in his article that the operators in the control room sometimes would be to occupied to answer telephones from the field operators. This would have been solved with a voice and read confirmation. This was debunked by the operator, who meant that it was rare that the control room did not pick up.

Another thing the operator pointed out was that he felt that communication consisting of only written messages would ruing the personal contact and, and the impressions the receiver would get from the senders tone of voice. He felt that if looking solely on the written message it is not always easy to separate what is an emergency and what is only information. This is the easiest with face to face conversation, but easier with phone calls than only written.

Chapter 6

Discussion

In chapter 3 there is a system suggestion that was approved as the prototype to be implemented. Some of the discussion, mainly which features that where selected to be implemented is placed there.

As stated in section 2.1 the development of a system involves an analytical part and an experimental part. The analytic's "What" part, as seen in figure 2.1, is taken care of in chapter 3. The "How"-part of this is also partly taken care of in that chapter with a suggestion for the layout. The construction of the software was however not explained until chapter 4, and discussed in chapter.

When selecting an internet protocol to use, the TCP was selected. Even though this part of the program is not going to be part of the program further, it turned out to be a good choice. Even though connecting to a computer, with an IP-address and port-number, may seem a bit complicated, this can not be solved unless one has a server to connect to. This server would be the server-side of the code and both the control room and the field operator would connect to this one. This requires a server that delivers a static IP-address, but is not hard to obtain. The use of the TCP-protocol worked perfectly fine for its purposes. The system needed stability, and especially the importance of having the messages delivered in the correct order is important to a chat application. As the TCP-protocol delivers both these features, it would be better than the UDP. When adding multimedia the UDP-protocol is a good choice as well, but when security is such an important aspect of the application the need for steady video streams would probably be less important. The videos and sound that is a possible solution for this application would not be streaming sound or videos either, but recording only to be sent and played on the portable device. The TCP-client in the applications was placed in a thread to keep the listening running even though the activity was recycled, as explained in section 4.11.

One of the main reasons for implementing a chat application to a power plant would be to be able to reach the field operators even in areas that they cannot

hear the broadcast calls. If deploying a system like this one would need to make sure that the internet connection is available at all places inside the plant.

Control Room, Desktop Program

The control room side program was implemented only to display the messages sent from the portable device, and check that the internet protocol actually worked. It also worked as a way of sending the read confirmation. This was implemented so that every time the pointer entered the frame, a read confirmation was sent to the portable device. The drawback of this is that it is not actually a read conformation more than it is a seen-confirmation. The operator sitting in the control room can for example only want to close the window, but even though s/he might not have seen the incoming message it will be sent a read confirmation.

This can be solved with adding another button to the view, where the operator in the control room needs to actively send the read confirmation rather than it being sent this automatically. Other listeners available that could fit were the focus listener and the window listener. The mouse listener could have been switched with the focus listener such that the operator would have to have interactions with the window, to send the listener. It can also be done with the mouse listener as well, combining the *mouseEntered()*, with the *mouseExited()* and the *mouseClicked()*. Then the operator would have to click the mouse inside the view to send the confirmation.

If to much of the operators work happen automatically he will no longer have the need to pay as much attention to what is happening, as seen in section 2.2. The operator in the control room will not need to do more than skim through the text without doing more than looking for buzz words that catch their attention. A positive thing of having the read confirmation sent automatically is that it is easy to forget to press the read-button and the field operator would be left wondering if the control room have seen his/hers message. S/he would then be needed to call, and the total process of one simple message would have been taken double the time than it should. As the control room operator would have as much need to see if the field operator have seen the message, as vice versa, the desktop program that are going to be implemented would have to have a way of displaying this.

Portable Device, Application

The application on the portable device was more advanced than the desktop program, and this application was to be incorporated into the existing software. The layout that was selected for the main chat view was a linear view. This view would display all items added to it in a sequential manner. The benefit of the linear layout is that everything places them selves in order, and the list view

can get the attention it needed. This would also have been obtainable with a relative view, where each element is placed in relation to the parent or other elements in the view. When testing the other views however they were not as easily controllable as the linear view.

The first change that was made to the program compared to the suggested system was to add the name of the sender to each chat item. One could easily separate the messages from the control room and the field operator in the current view, but if more than two parts where to be added to the conversation it would immediately be a bit harder. The name of the sender is therefore added to the top of each item. The message it self is then added underneath it, with a bigger font. Each item is rather large and it should not be a problem to read the message.

The relative view was considered not fitting for the complete chat view; however it was used when implementing the task bar in the bottom of the view. This task bar contained in the beginning only the simple edit text field, and the send button. After some time this layout was changed. The background colour of the bottom bar was made to match the same the top bar. These changes were made to add user friendliness to the application as well as a sense of a professional application in stead of a home made one. In addition is was made like this to resemble the commonly known applications from section 2.5, and when the layout is similar to something one has seen before, one will sooner perceive it as familiar, and the level the operator bases his/hers decisions will faster be raised to a higher level. This can be tested in the upcoming trial, with a form to be filled out after the test rating the system on familiarity and user friendliness.

When adding the button to trigger the selection of the phrases, the button was placed inside the edit text field. As two buttons were to be added it felt as the best system choice to add the button that would add an item in the edit text field inside that field. Seen in the applications from 2.5 the photography button is often placed on the side of the text field.

Pressing this plus button would open a list of categories. These categories was selected in 5.1. The original idea was that these categories were to displaying categories like "Work task", "Confirmation", "Maintenance" etc. With the lack of papers describing the communication it was very hard to come up with a set of phrases that would be general enough to be applied on a general basis. The suggestion for these phrases would then be that the phrases were divided by which system they belonged to. As the NPP is divided into different areas of responsibility, this would mean that each field operator generally would only need one of these categories. Still after suggesting splitting the phrases into system parts, it was difficult to determine specific phrases. It was even confirmed by the operator in section 5.2 that the span of the information is too wide to determine a working set in advance. This would be even harder as the author have no real life experience with the way the communication works in a plant. To determine these phrases, the most logical thing to do is to let each field operator add his/hers own categories and fill them with phrases they feel they

can apply frequently.

That the field operators their selves could add new content was taken into consideration already when implementing the system. The main drawback of adding phrases as the system is today is that it is not possible to add more than one phrase to a category at the time. Sending information between actions is a rather static affair, and one can not change the number of items one send. The thing that could have been changed was that one added the data to an arrayList, and therefore could send as many phrases as one would desire. As this feature was added before the save-function, it was not considered at the time to add a save-variable that saved the added content directly to the SD-card without returning to the category activity. When implementing the adding of new content several methods were tried out. The first thing tested was a view switcher that was called every time the user held the lowest button for a time. This would change the view from only text to an edit text with a add button. The drawback of this was that the switcher was called and inflated the view every time the view was touched. Even though it would add a new item the system would keep the view switch on the selected item, which by the time would have been the second lowest item. When this problem was resolved the problem began that every time one wanted focus to the edit text area the getView function inflated the entire view over and over, so that one would never be able to add any content. After this failed trial, an idea was to add an intent that would simply show in front of the text, in a similar manner as the picture selector. This intent is a finished feature from Android, and a fitting element was not found. Therefore the solution of a new activity was added. The solution may not be as neat as the other two, but it is more solid, and the possibility of unstable releases on this is very slim.

As mentioned earlier the intent for choosing a photo was available and the button was placed to the right of the edit text field. When adding pictures, it was a possibility of adding videos and sound files as seen in other chat applications. This would increase the load on the connection and the system as the operator system would need to buffer for every lost segment. As the control room have constant supervision of the plant, it is no much they need a video of that they cannot see in their own panels. It was considered as enough to add pictures to the chat. The addition of sound files was regarded as unnecessary as there is a possibility of using the landlines for contact if one would like to send a message to the control room.

Another feature that was added to the chat application especially for purposes in an industrial environment was the sorting of messages concerning one item. This would accord section 2.4.1 avoid confusion in an ongoing chat. With the massive amount of units in a NPP displaying messages only regarding one item can possibly display nothing of interest for the operators. To change this to a group will probably be a better idea, and will display information one could put into use to find related problems. The prefix sorted on did seam like a good idea, and is not used by any other typical communication statements in a NPP.

The sorting of the messages would however sort the problem with the extensive amount of messages. After a while the amount of messages would be rather big, and this sorting would make it easier to locate earlier message one would want to take a look at.

After adding the possibility of sorting the messages on ids, it became clear that the amount of arrayLists became rather large. This could have been avoided by making more arrayLists containing arrayLists. Unfortunately it is no other way to display information in a listView, and one therefore needs the lists. This is one of the major pitfalls of the code, as the possibility of, when one is extending the code, forgetting to add to one of the arrays would cause a crash of the application.

The final improvement of the program was to add the opportunity of saving the messages locally. When saving the messages, another person that use the device in a later time would easily see what have been done from before. When seeing this together with the extensive monitoring from the control room it could however bee seen as some sort of surveillance. When an operator feels that s/he is being monitored at all time, it have been registered that they experience a higher rate of stress[22]. The rate of stress with and without the storing of messages is something that needs to be tested for in the trial. As there was no server available to store the chat items on, the stored messages was stored locally. The possibility of saving messages when they are received also introduces the opportunity of using the system as an asynchronous message system as long as the device has internet connection.

When making the system suggestion in chapter 3 the possibility of adding timer to make sure that it did not arrive to many messages at once was removed. This was in the system suggestion regarded as a non-existent problem in the suggestion. It should however be tested in the trial. If having a group-conversation between several field operators and several persons in the control room could result in many messages. In section 2.4.1 they had added a timer that made sure that each message would get a certain time limit where that message was the new message. This would cause a queue, and the argument from chapter 3 was that this could block important messages. A lot annoyance with a system would however lead to the users not applying it as much as the makers would have like them to, and should be avoided. To avoid both the flood of messages it is possible to add a timer, and in addition add a priority index to each message. These priority indexes should be determined beforehand, and would make important messages skip the queue.

Interview with Operator

After the prototype was implemented it was shown to an operator. The writers of the system in section 2.4.1 sat down with the potential users of the program and decided on phrases that would be useful and this meeting was originally

meant to be helped to develop a set of pre-stored phrases. It did however develop into a more general conversation about the application. As the experimental phase of this prototype will not be done until the complete application is finished, this was as close to an evaluation of the prototype one could come. However the system, approved by scientists, was not very well received by the operator. With the benefit of hindsight this interview should probably have taken place earlier in the process, maybe as early as the approving of the system suggestion. This would have made it easier to implement the changes that was suggested in chapter 5. In that way one could have altered more between stages in figure 2.1. It also need to be taken into consideration that it might be some misunderstandings in the explanation of the system, making it unclear how it would work, and that this is the opinions of one man.

One of the suggestions was to change the saved phrases to a selection of ids. This can easily be changed. The source resource folder in the project contains a text file that can easily be altered. If one wishes to make this change, one can keep the same system categories and replace the phrases with the ids in the same part of the process. Mentioned in section 2.4.1 a drawback with the ids is the wrongly marked messages. When adding the lists of ids it might help to lower the percentage of the errors in this area. This is something that needs to be tested in the trial though. The trial could test both alternatives, with the phrases and the ids, and see which system they find most useful. They could then use the selected phrases to add the phrases they use the most their selves.

The amount of mistakes in a conversation in a nuclear power plant needs to be virtually none. It was explained by the operator that those few error that occur in oral communication normally gets corrected at once. Stated by Rognin[32] is that cooperation between operators with closely related work tasks will in most cases team members are rather tolerant of mistakes from other team members. This is in a situation in air control where the potential faults may not be as severe as the most extreme faults of a power plant, but still rather severe. The same mistakes obtained by vocal communication can exist within written communication as well, and it is reasonable that if the operators have a high degree of mutual awareness this will also be discovered and corrected in a written message.

In section 2.4.3 a social communication tool was added to an industrial environment, and tested there. The result of this trial was that 30% of the workers in the test-project used the new method of communication during the test trial. This was the number of workers that have tested the system one or several times during the whole trial of six months. When this is taken into account, 30% is not an impressive number. When adding that it was voluntarily to use it, it makes the percentage a little more decent. It still indicates that it might not only be safety that is a factor when renewing a rather conservative industry. When workers have worked a while and, have reached the skill based action pattern as seen in section 2.2, they might not be as willing to go back to the knowledge based level. The operator in the interview was himself more than

fifty years old, and on a general basis this would mean that he would not be as familiar with chatting on portable devices as per example a person of twenty years. This can be seen with the argument given that written communication is not as personal as a phone call, although it might not mean that much.

Another argument is that when introducing the aspect of new technology to a conservative business, the people who work there might also be a bit conservative. With a system that can cause severe damages to the environment if something goes wrong, they might feel reluctant to change the level of automation or procedures because if it works, they do not see the point in changing it at all. The early version of Murphy's Law in section 2.2 explained that one should be hesitant to add new elements into an already complex system. Attention obtained by the operator should be in a manner that the operator would handle it when s/he have time for it. As stated by Handel and Herbsleb[3] people that are inexperienced with instant messaging will be more insecure about it, and feel more distracted by the constant alarms that are popping up at the screen. Therefore a part of the trial should be to fill out a form in advance where one would rate how experienced one is with the instant messaging form of communication as well as how familiar one is with smart phones with touch screen and soft keyboards. To see how experienced the user is, compared to how much s/he uses the chat application, and the level of stress he feels when alarms of incoming messages appear would be useful for further development of the system, as it was experienced by Handel and Herbsleb that group chats was less disturbing than one-on-one conversations. This can be used to make the system fitted to each user. It is reasonable to believe that people that are more used to applying a similar system in their private life would be more inclined to use it in a professional manner and less disturbed by the alarms.

The article by Handel and Herbsleb[3] states, as seen in section 2.4.3 that during the day, communication between people revolve less about work and more about social or humorous messages. The users of this system will probably also be affected by this tendency. This can be tested in the trial, and the belief is that it might increase around the times for food, and otherwise kept on a rather strict base of work. As the communication between the control room an field operators from before contain rather strict informational messages or instructions, it is believed that this tone will be maintained in the written form of the messages. Messages among field operators working more closely together might stray a bit more.

Hypothesis

As a part of the assignment a set of hypotheses to be tested in the trial during 2014 needed to be formed. These are suggested throughout this chapter, and summarizes here. By testing a chat system there is not many numerical values that can be tested. Some numerical values that can be logged is:

- What is the ratio between the use of the chat-system and the landlines system.
- Number of misunderstandings with the chat application compared to the phone system.
- Number of wrongly marked ids with pre-stored compared to written.
- Which feature is most frequently used; pre-stored ids or pre-stored phrases.
- Rate of non-work related messages.

It is reasonable to believe that the use of landlines still will be more used than the chat application, especially with not that experienced users, if believing the indications from the operator interviewed. The number of misunderstandings is believed to be less than with oral communication as one can see what is stated at all time, and the number of wrongly marked ids with pre-stored are supposed to be less than those written freely.

Filling out forms in advance of the trial one can get numerical values also on the following statements;

- Stress level with surveillance compared to not monitored
- Stress level of more experienced users compared to beginners
- Ratios between chat system and landlines-system compared to the amount of experience the user have with other chat applications
- The loss of focus when an alarm appears, comparing experienced users compared to beginners
- The experience of user friendliness, comparing experienced users compared to beginners
- How positive is the person to changes, compared to the amount the feature is used?

From the earlier discussion it would be reasonable to believe that there should not be a higher stress level when being monitored than not being. It should be approximately equal. The stress level of more experienced users is believed to be less than for users that are not as familiar with the system, and the same for the loss of focus. It is also believed that persons using other chat applications in their private life prone to use the chat feature at work more.

If the experienced users feels the user friendliness is better than the inexperienced users it not as easy to determine in advance. One argument is that experienced users will easily adapt to similar systems as they have similar things before. Another argument is that as the experienced users will keep comparing the system to other systems they are using, and be caught up in things that are not the same. It is believed that if the user has some experience the user will improve to a higher level of behaviour faster than those who have not seen it. Making them feel a higher sense of user friendliness.

How positive the person is to changes compared would probably also influence how much the person use the system, and it is believed that change friendly persons will adapt to the possibility of the chat application faster than non-change-friendly.

Chapter 7

Conclusion

This Master's Thesis was written for the OECD Halden Reactor Project, and the assignment has been determined by IFE, Halden.

The work in this Master's Thesis consists of four parts, literature study, a system suggestion to be approved by IFE, a system implementation and a list of suggested hypothesis to be tested in the upcoming trial.

Portable devices have not been deployed to Nuclear Power Plants previously, and a chat feature is not a part of the current procedures of communication. Therefore literature from similar areas of work had to bee applied, and to the best extent adapted to the circumstances regarding power plants.

The system suggestion presented is a composition of ideas from IFE and tested ideas from the theory chapter. The layout of the application in the system suggestion is developed looking at commonly known Android communication applications, to obtain a layout as familiar as possible for the user. This would according to theory make the users more inclined to sooner be acquainted with the chat, and improve user friendliness. It was also suggested to add some features to avoid confusion and misunderstandings in the communication. Additions to avoid confusion from system suggestion are read confirmation, saved phrases and log of messages regarding one item. This system suggestion was displayed to scientists at IFE, and approved as the system to be implemented.

When implementing the application the system suggestion functioned as a foundation and was followed to a large extent. Some changes was made to the application to accommodate for programmatic challenges, and to better the perceived user friendliness and the entirety in the application.

After the application was finished, it was made an attempt on making categories and filling them with phrases. This turned out to be difficult with the amount of experience in communication between control room and field operators, combined with the lack of articles on the subject. To add more experience

a operator was interviewed. He did not believe in the feature of phrases as it was too many items in a power plant to talk about, and it was difficult to generalize conversations. Subsequently he did not believe that messages should be sorted by ids either. It is a possibility of sorting the ids on the systems. The operator believed mostly in the possibility of adding multimedia messages to the conversation.

This existing system is still in the early stages of testing. The resulting application of this Master's Thesis is being used as an input to the chat application included into the portable units' project that is to be tested in the virtual laboratory in Halden summer 2014.

Despite the scepticism from the operator regarding the use of chat-feature, it is still regarded as a positive input to improve efficiency by scientists. The system should therefore be tested in the upcoming trial to get results and opinions from several operators and not just one. The operators should ideally have distributed ages, and experience levels with chat from a private setting. A suggested list of hypothesis has been compiled with the most important being:

- The landlines system will still be more used than the new chat feature.
- Change-friendly persons will be more inclined to use the chat feature.
- Persons with extensive experience of chat applications in private will use the chat feature more frequent than people not using other chat applications.
- There will be fewer misunderstanding with the chat feature, than with vocal communication.
- Stress level of experienced users will be less than for those less experienced.
- The perception of user friendliness will be higher for experienced users.

Bibliography

[1] C. Skourup, "Ttk1 opperatørkommunikasjon, lecture slides," December 2012.

[2] P. V. R. Carvalho, M. C. R. Vidal, and E. F. de Carvalho, "Nuclear power plant communications in normative and actual practice: A field study of control room operators' communications: Research articles," *Hum. Factor. Ergon. Manuf.*, vol. 17, pp. 43–78, Jan. 2007.

[3] M. Handel and J. D. Herbsleb, "What is chat doing in th workplace," *CSCW '02 Proceedings of the 2002 ACM conference on Computer supported cooperative work*, 2002.

[4] Technocreeps. `http://technocreeps.com/wp-content/uploads/2013/04/facebook-messenger.jpg`. Online, accessed 17.11.2013, 15:44.

[5] Google, "Facebook." `https://play.google.com/store/apps/details?id=com.facebook.katana`. Online, accessed 17.11.2013, 15:57.

[6] Google, "Viber." `https://play.google.com/store/apps/details?id=com.viber.voip`. Online, accessed 17.11.2013, 19:12.

[7] Google, "Skype." `https://play.google.com/store/apps/details?id=com.skype.raider`. Online, accessed 17.11.2013, 19:29.

[8] Google, "Hangouts(erstatter talk)." `https://play.google.com/store/apps/details?id=com.google.android.talk`. Online, accessed 17.11.2013, 20:00.

[9] Google, "Snapchat." `https://play.google.com/store/apps/details?id=com.snapchat.android`. Online, accessed 17.11.2013, 19:47.

[10] A. Developers, "Activity." `http://developer.android.com/reference/android/app/Activity.html`. Online, accessed 22.11.2013, 14:42.

[11] M. CAD, "Marshall's cad drawings." `http://www.oocities.org/mbegel/drawings.htm`. Online, accessed 05.12.2013, 21:05.

[12] nixCraft, "What is the difference between udp and tcp internet protocols?." `http://www.cyberciti.biz/faq/`

key-differences-between-tcp-and-udp-protocols/. Online, accessed 16.11.2013, 30:37.

[13] GSMArena, "Samsung i9300 galaxy siii." http://www.gsmarena.com/samsung_i9300_galaxy_s_iii-4238.php. Online, accessed 30.11.2013, 15:07.

[14] Clove, "Samsung galaxy s3-blue." http://www.clove.co.uk/samsung-galaxy-s3-blue. Online, accessed 08.12.2013, 12:48.

[15] Øycind Stavdahl, "Systemutvikling med uml, lecture slides." http://www.itk.ntnu.no/fag/TTK4125/2013/index.php?site=lysark, 2013.

[16] E. N. Karsten Bråthen and A. C. Jenssen, "Utvikling av menneske-maskin-systemer," *FFI-report*, 2001.

[17] G. R. Jr., "The pros and cons of automation," *Scribd*, 2009.

[18] A. B. Aune, *Brukerkommunikasjon i automatiserte anlegg*. Trondheim, Norway: Department of Engineering Cybernetics, NTNU, 2000.

[19] A. D. Society, "Murphy's law in english." http://listserv.linguistlist.org/cgi-bin/wa?A2=ind0710B&L=ADS-L&P=R432&I=-3. Online, accessed 28.11.2013, 14:07.

[20] W. Johnson, *MORT Safety Assurance Systems,*. New York, USA: Marcel Dekker Inc., 1980.

[21] T. B. S. Raja Parasuraman and C. D. Wickens, "A model for types and levels of human interaction," *Unknown*, 2000.

[22] T. Onshus, *Instrumenteringssystemer*. Trondheim, Norway: Department of Engineering Cybernetics, NTNU, 2011.

[23] B. K.-H. Sun and D. G. Cain, "Computer applications for control room operator support in nuclear power plants," *Reliability Engineering and System Safety*, vol. 33, no. 3, pp. 331–340, 1991.

[24] P. V. D. Carvalho, "Ergonomic field studies in a nuclear power plant control room," *Progress in Nuclear Energy*, vol. 48, no. 1, pp. 51 – 69, 2006.

[25] H. Svengren, "Kommunikastion," *Internal document, IFE*, 2013.

[26] O. Citionary, "Definition of chat in english." http://www.oxforddictionaries.com/definition/english/chat. Online, accessed 28.11.2013, 13:37.

[27] D. Gonzalez, "Teaching and learning through chat: A taxonomy of educational chat for efl/esl," *Teaching English With Technology, vol.3*.

[28] A. dEca, "To chat or not to chat in the efl classroom.," *SLanguage – Communication - Culture, International Conference, University of Evora, Portugal, November 29*, 2002.

[29] M. Webster, "chat." http://www.merriam-webster.com/dictionary/chat. Online, accessed 28.11.2013, 15:07.

[30] I. A.D.Stewart and P. File, "Let's chat: A conversational dialogue system for second language practice," *Routledge-Online*, 2007.

[31] C. J. P. d. L. Mariano Pimentel, Hugo Fuks, "Interactivity in the context of designed experiences," *Journal of interactive advertising*, vol. 1, p. 75, 2000.

[32] L. Rognin and J.-P. Blanquart, "Human communication, mutual awareness and system dependability. lessons learnt from air-traffic control field studies," *Reliability Engineering and System Safety*, vol. 71, no. 3, pp. 327 – 336, 2001.

[33] M.Slatella, "The office meeting that never ends," *New York Times*, 1999.

[34] T. Koskinen, "Social software for industrial interaction," *Ozchi.org*, 2006.

[35] Wikipedia, "Facebook." en.wikipedia.org/wiki/Facebook. Online, accessed 17.11.2013, 16:00.

[36] Wikipedia, "Viber." http://en.wikipedia.org/wiki/Viber. Online, accessed 18.11.2013, 15:42.

[37] Wikipedia, "Skype." http://en.wikipedia.org/wiki/Skype. Online, accessed 18.11.2013, 16:09.

[38] Wikipedia, "Snapchat." http://en.wikipedia.org/wiki/Snapchat. Online, accessed 18.11.2013, 13:43.

[39] Cnet, "Snapchat snapshot: App counts 8m adult users in u.s.." http://news.cnet.com/8301-1023_3-57590968-93/snapchat-snapshot-app-counts-8m-adult-users-in-u.s/. Online, accessed 18.11.2013, 14:25.

[40] A. Developers, "Building your first app." http://developer.android.com/training/basics/firstapp/index.html. Online, accessed 14.11.2013, 14:44.

[41] A. Developers, "Creating an android project." http://developer.android.com/training/basics/firstapp/creating-project.html. Online, accessed 14.11.2013, 15:23.

[42] S. Developers. http://developer.samsung.com/android. Online, accessed 14.11.2013, 15:35.

[43] J. C. Geeks, "Android socket example." http://examples.javacodegeeks.com/android/core/socket-core/android-socket-example/. Online, accessed 14.11.2013, 16:23.

[44] T. Android, "Incorporating socket programming into your applications." http://thinkandroid.wordpress.com/2010/03/27/

incorporating-socket-programming-into-your-applications/. Online, accessed 14.11.2013, 16:23.

[45] A. Solutions, "Android tcp connection tutorial." http://myandroidsolutions.blogspot.no/2012/07/android-tcp-connection-tutorial.html.

[46] Edumobile, "Socket programming." http://www.edumobile.org/android/android-development/socket-programming/. Online, accessed 14.11.2013, 16:23.

[47] A. Developers, "Building a simple user interface." http://developer.android.com/training/basics/firstapp/building-ui.html. Online, accessed 15.11.2013, 19:14.

[48] M. D. Network, "How to: Create an inbound rule in windows firewall for the port of microsoft dynamics nav web client." http://msdn.microsoft.com/en-us/library/hh168549(v=nav.71).aspx. Online, accessed 16.11.2013, 12:39.

[49] A. Developers, "Intent." http://developer.android.com/reference/android/content/Intent.html. Online, accessed 05.12.2013, 12:49.

[50] Wikipedia, "A picture is worth a thousand words." http://en.wikipedia.org/wiki/A_picture_is_worth_a_thousand_words. Online, accessed 21.11.2013, 18:37.

[51] A. Developers, "Installing the eclipse plugin." http://developer.android.com/sdk/installing/installing-adt.html. Online, accessed 06.12.2013, 16:28.

Appendix A

Example of Communication Between Field Operator and Control Room

This section will display examples of communication within a Nuclear Power Plant.

Turbin **O**perator	An alarm on low coolant on 441 PC2 has arrived. I want you to open 441 VC102 a bit so that the alarm goes away. You can take a look at the local flowmeter 441 KC316 when you adjust the flow. Call me when you have adjusted the flow.
Field **O**perator	I will open 441VC102, so that the flow alarm goes away. I will call when this is done.
TO	Good, that is correct.

Table A.1: Communication, Example 1

Reactor **O**perator	We have received a fault report on a somewhat elevated exhaust temperature on a cylinder on Diesel A, when test-running it yesterday. We are therefore doing a new test now, and the mechanics department are doing measurements as we go along. I want you to do controls before the test drive, and call me when you are finished.
Field **O**perator	I will do controls before test run on diesel A. I will notify you when I am finished. Thereafter mechanics are going to join when test running it to control the cause of the elevated exhaust temperatures on a cylinder.
RO	Yes, that is correctly perceived. How long do you think before you are finished?
FO	I need to finish the coating on the 324-filter first. I am almost finished. In 45 minutes I should be ready to start diesel A.
RO	Very well. Finish 324 first. I will notify mechanics that we can start the trial run in about 45 minutes
FO	OK

Table A.2: Communication, Example 2

Appendix B

System Specifications, Samsung Galaxy S3

GENERAL	2G Network	GSM 850 / 900 / 1800 / 1900
	3G Network	HSDPA 850 / 900 / 1900 / 2100
	SIM	Micro-SIM
	Announced	2012, May
	Status	Available. Released 2012, May
BODY	Dimensions	136.6 x 70.6 x 8.6 mm (5.38 x 2.78 x 0.34 in)
	Weight	133 g (4.69 oz)
DISPLAY	Type	Super AMOLED capacitive touchscreen, 16M colors
	Size	720 x 1280 pixels, 4.8 inches (306 ppi pixel density)
	Multitouch	Yes
	Protection	Corning Gorilla Glass 2 - TouchWiz UI
SOUND	Alert types	Vibration; MP3, WAV ringtones
	Loudspeaker	Yes
	3.5mm jack	Yes
MEMORY	Card slot	microSD, up to 64 GB
	Internal	16/32/64 GB storage, 1 GB RAM
DATA	GPRS	Class 12 (4+1/3+2/2+3/1+4 slots), 32 - 48 kbps
	EDGE	Class 12
	Speed	HSDPA, 21 Mbps; HSUPA, 5.76 Mbps
	WLAN	Wi-Fi 802.11 a/b/g/n, dual-band, DLNA, Wi-Fi Direct, Wi-Fi hotspot
	Bluetooth	Yes, v4.0 with A2DP, EDR

	NFC	Yes
	USB	Yes, microUSB v2.0 (MHL), USB On-the-go
CAMERA	Primary	8 MP, 3264x2448 pixels, autofocus, LED flash, check quality
	Features	Simultaneous HD video and image recording, geo-tagging, touch focus, face and smile detection, image stabilization
	Video	Yes, 1080p@30fps, check quality
	Secondary	Yes, 1.9 MP, 720p@30fps
FEATURES	OS	Android OS, v4.0.4 (Ice Cream Sandwich), upgradeable to 4.3 (Jelly Bean)
	Chipset	Exynos 4412 Quad
	CPU	Quad-core 1.4 GHz Cortex-A9
	GPU	Mali-400MP
	Sensors	Accelerometer, gyro, proximity, compass, barometer
	Messaging	SMS(threaded view), MMS, Email, Push Mail, IM, RSS
	Browser	HTML, Adobe Flash
	Radio	Stereo FM radio with RDS
	GPS	Yes, with A-GPS support and GLONASS
	Java	Yes, via Java MIDP emulator
	Colors	Pebble blue, Marble white, Amber brown, Garnet red, Sapphire black, Titanium grey, La Fleur - S-Voice natural language commands and dictation - Smart Stay eye tracking - Dropbox (50 GB storage) - Active noise cancellation with dedicated mic - TV-out (via MHL A/V link) - SNS integration - MP4/DivX/XviD/WMV/H.264/H.263 player - MP3/WAV/eAAC+/AC3/FLAC player - Organizer - Image/video editor - Document viewer (Word, Excel, PowerPoint, PDF) - Google Search, Maps, Gmail, YouTube, Calendar, Google Talk, Picasa - Voice memo/dial/commands - Predictive text input (Swype)
BATTERY		Li-Ion 2100 mAh battery
	Stand-by	Up to 590 h (2G) / Up to 790 h (3G)
	Talk time	Up to 21 h 40 min (2G) / Up to 11 h 40 min (3G)

MISC	SAR US	0.55 W/kg (head) 1.49 W/kg (body)
	SAR EU	0.21 W/kg (head)
	Price group	About 280 EUR
TESTS	Display	Contrast ratio: Infinite (nominal) / 3.419:1 (sunlight)
	Loudspeaker	Voice 75dB / Noise 66dB / Ring 75dB
	Audio quality	Noise -90.3dB / Crosstalk -92.6dB
	Camera	Photo / Video
	Battery life	Endurance rating 50h

Table B.1: Samsung S3 System Specifications[13]

Appendix C

How to Install Content on CD

The CD in the back of this assignment contains two .zip files. One named DesktopProgram, which is the Java program with the program for the desktop, and one named PortableUnitApplication, which is the code for the application to run on the portable unit. This chapter is an user manual for how to start using this code.

C.1 Desktop Program

To develop Java code, one need a text editor and a compiler. In this example the Eclipse editor is shown, and serves as a very good option when developing for Java. This can be downloaded at eclipse.org, and has a compiler included in its installer.

The first thing to be done when the editor is installed is to go to "File", and select "Import", as shown in C.1.

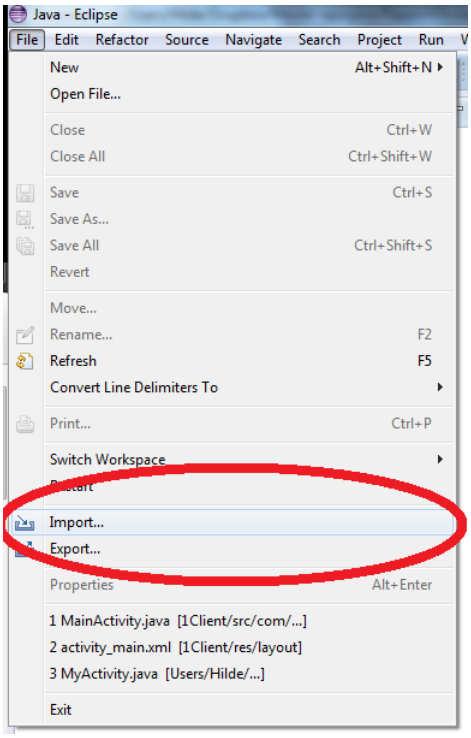


Figure C.1: Import Project

When pressing this option a pop-up window as in figure C.2 will show. Select **Existing Projects into Workspace**, situated in the **General**-folder. It may seem as a good idea to press the **Archive File**-option. This will not work, as this option is if you are importing Eclipse resources into an existing Eclipse project. As you are most likely importing the entire project the **Existing projects** is the correct option.

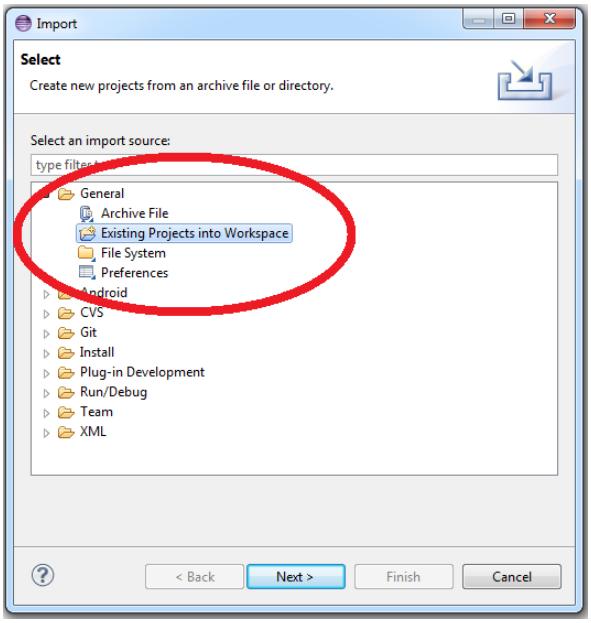


Figure C.2: Importing Existing Projects

If you have clicked on the correct buttons the window that pop up should look somewhat similar to figure C.3. Click the radio-button next to **Select archive file** and then select **Browse**. Find the archive file on the CD, and then click **Open** to select it. If the file is correctly zipped, the project-name will appear with a check mark in the box in front of it. Then select **Finish**, and the project should be situated in your workspace.

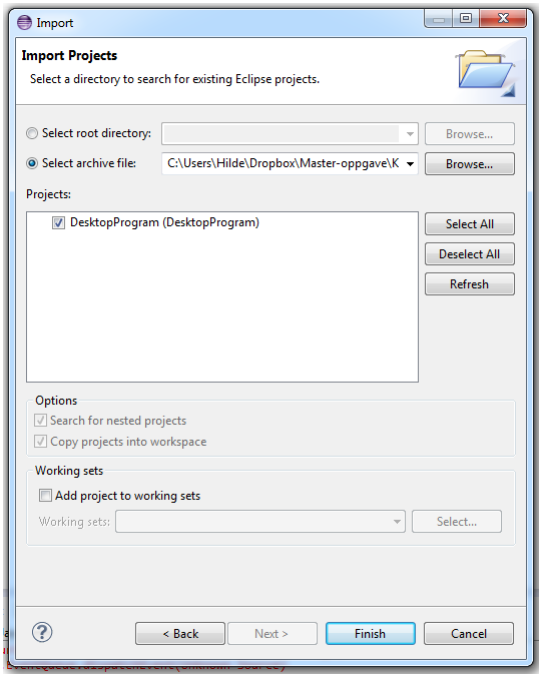


Figure C.3: Importing Archive file

C.2 Portable Unit, Application

When developing an Android application one also need a text editor and a compiler. If you are using the Eclipse editor, it is easy to get started with this. The editor is already in place, and then one need to follow three simple steps;

- Download the Android SDK.
  Provides the libraries and developer tools necessary to build, test and debug apps for Android. Contains:
  - Eclipse ADT plugin
  - Android SDK Tools
  - Android Platform-tools
  - The latest Android platform
  - The latest Android system image for the emulator
- Install the ADT plugin for Eclipse[51]
  1. Start Eclipse, then select **Help** > **Install New Software**.

2. Click **Add**, in the top-right corner.

3. In the Add Repository dialog that appears, enter "ADT Plugin" for the Name and the following URL for the Location: https://dl-ssl.google.com/android/eclipse/

4. Click **OK**.
   If you have trouble acquiring the plugin, try using "http" in the Location URL, instead of "https" (https is preferred for security reasons).

5. In the Available Software dialog, select the checkbox next to Developer Tools and click **Next**.

6. In the next window, you'll see a list of the tools to be downloaded. Click **Next**.

7. Read and accept the license agreements, then click **Finish**.

8. If you get a security warning saying that the authenticity or validity of the software can't be established, click **OK**.

9. When the installation completes, restart Eclipse.

When Eclipse restart, do the following to configure the plugin;

1. In the "Welcome to Android Development" window that appears, select Use existing SDKs.

2. Browse and select the location of the Android SDK directory you recently downloaded and unpacked.

3. Click Next

- Download the latest SDK tools and platforms using the SDK manager. This is added with the installation of the SDK.

When running the application one need to have a portable unit available through the USB-port, or make an Android Virtual Device. The AVD-manager was installed along with the Android SDK, and can be made in this. There are a lot of great guides to this on the internet.

If you are using a real portable unit you need to activate your android device for debugging. That is the slid-button on top, and the USB-debugging option in figure C.4.

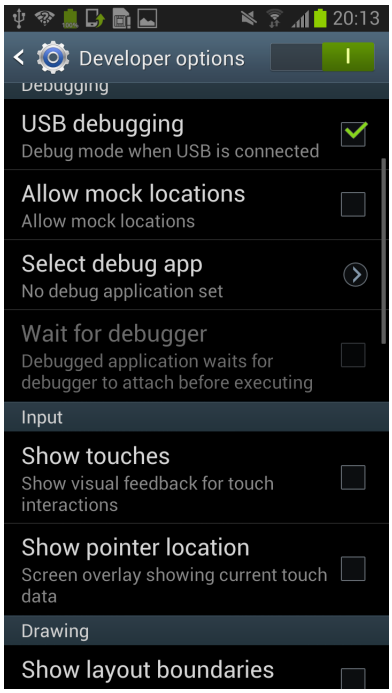


Figure C.4: Debugger Options in Android Phone

When importing the content on the CD, the procedure is the same as in C.1.

Appendix D

User Manual

D.1 Desktop Program

This section contain a user manual for the implemented desktop programs. Figure D.1 shows the different items in the program.

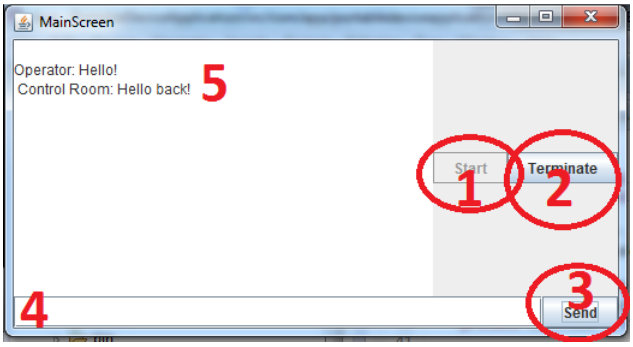


Figure D.1: Desktop Program

1. Before starting the application the start button need to be clicked. This make the server listen for connections from the client.
2. This button terminates the connection, and closes the socket. To be able to reconnect on the same socket, it is important to remember pressing this button.
3. The send button send the message in the input area.
4. This is the input area for the messages.
5. Here the sent and received messages will be displayed.

D.2 Portable unit

This section contain a user manual for the implemented application made in this Master's Thesis. It is important the the control room program is started before starting the portable device application. The main view is as follow in figure D.2. When clicking the items inside the circles, the view that is triggered that have the same list number as the number inside the circle.

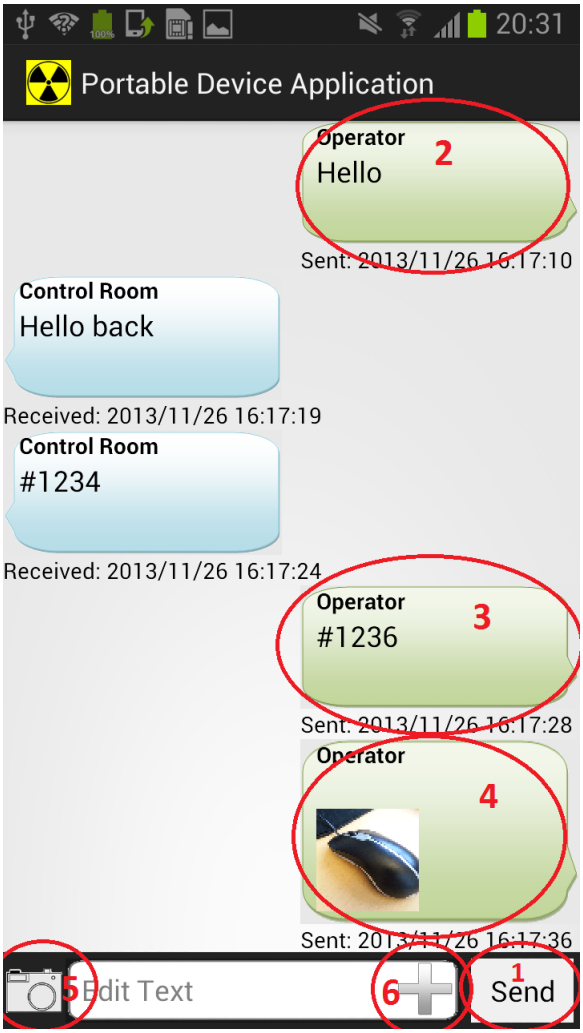


Figure D.2: Clickable items in main view

1. **Send a message**
   To make a message, one need to press the "Edit Text"-field. This activates the soft-keyboard as seen in figure D.3. After typing the message, the button inside circle 1 need to be pressed. This will send the message to the server and add it to the chat view.

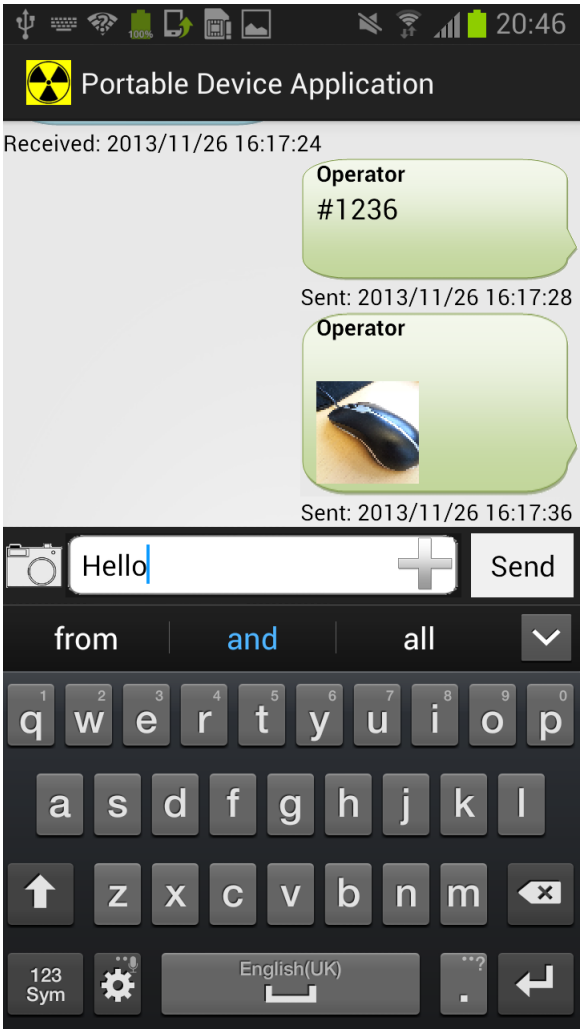


Figure D.3: User Manual Sending message

2. **Pressing chat item**
   When pressing a chat item in the chat view which is only text, this will trigger the activity shown in D.4. This displays the message it self in a bigger font, for accommodating for readability.

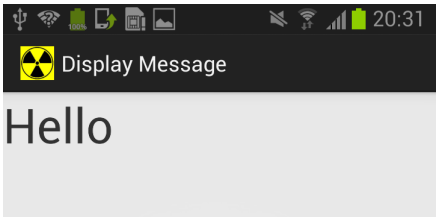


Figure D.4: User Manual Pressing Normal Chat Unit

3. **Pressing chat item with id**
   When pressing a chat item with a ID-number inside, like the chat item inside circle 3, the activity loaded is as seen in figure D.5. This checks which ID's is in the chat item pressed, and return a view of all items with same ID. This also works when several ID's is involved.

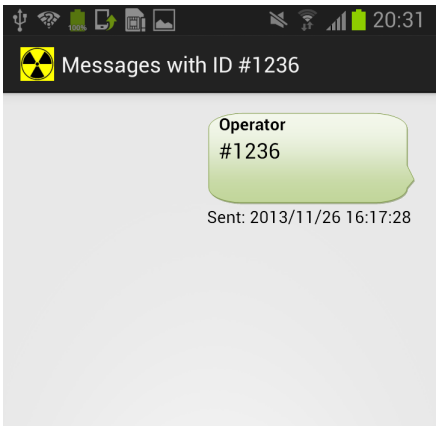


Figure D.5: User Manual See Messages With ID

4. **Holding chat item with picture**

   When messages is containing a picture, this picture is set to thumbnail. Although one can see what the picture is displaying, it is not easy to see the details. To see the picture in full screen, one need to hold the chat item that are displaying the picture one want to examine. When holding a chat item, like the one inside circle 4, a view like the one in D.6 will appear.

Figure D.6: User Manual Display Chat Picture

5. **Adding picture** To be able to add, and send a picture, to the Control Room one need to click the button inside circle 5. If you want to add a message to the picture being sent, the one need to add the messages in the "Edit Text"-field before pressing this button. When pressing the camera button, the intent shown in figure D.7 will appear. To take a new picture, press camera, and to upload a stored picture, press Gallery. This will send the picture to the control room and add the picture to the chat view.

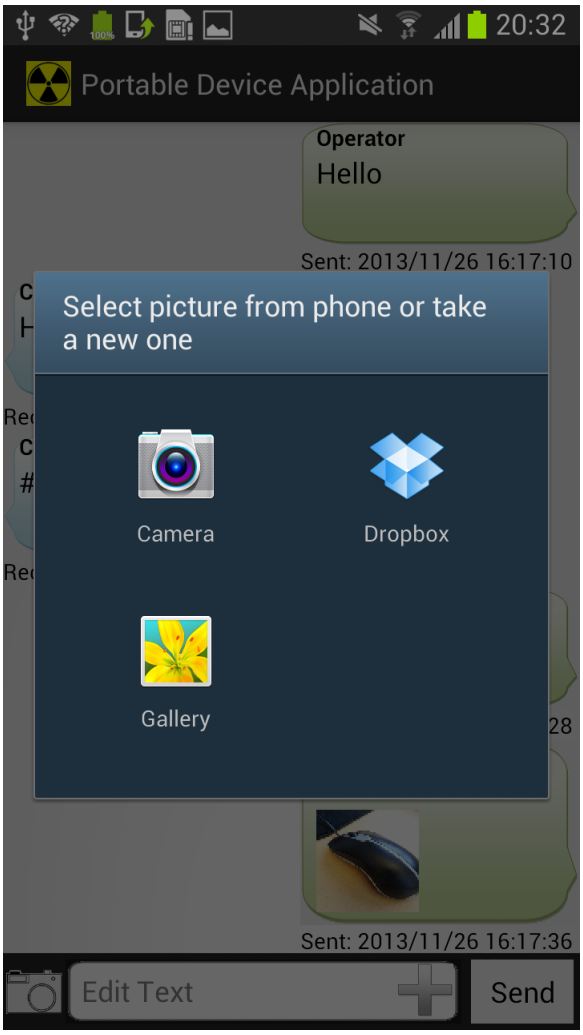


Figure D.7: User Manual Sending Picture

6. **Adding saved content** When pressing the plus-button inside circle 6 the activity displayed in figure D.8. This activity displays the different categories that expressions can be selected from.

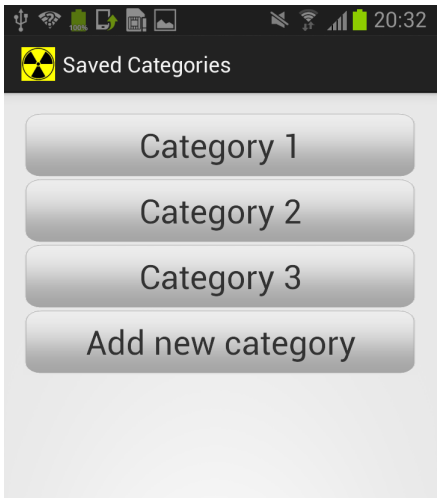


Figure D.8: User Manual Pressing Pluss

When deciding on an category, you press that activity. If one per example press Category 1, the activity displayed in figure D.9. When pressing one of these buttons the content of this button will be added to the Edit Text field in figure D.2. This content is then editable, and one can send in like shown in list item 1.

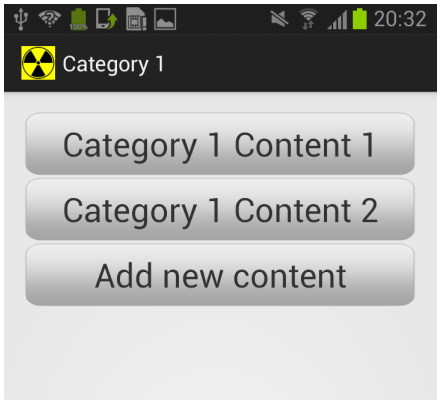


Figure D.9: User Manual Pressing Category

To be able to add a new category one hold the button with "Add new

category". The activity in figure D.10 will the be shown, and the wanted name of the new Category can be added. This will add the category to the category list, and make a new list with only a "Add new content"-item.

To add new content to a category list one can hold the bottom button in the content list. The activity in figure D.10 will then be loaded, and the new content can be added. It is important to only add one new content before returning to the category list. If not, only the latest added content will be added permanently to the list.

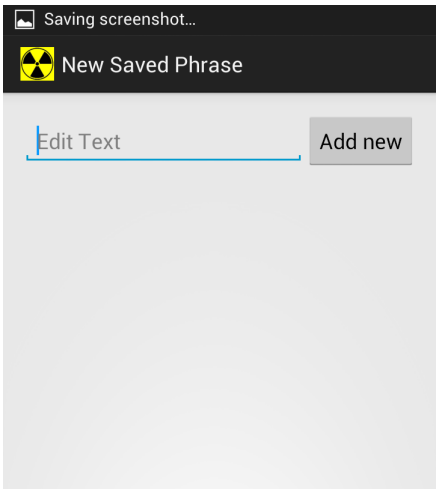


Figure D.10: User Manual Adding Category or Content

Appendix E

Entire system

This chapter is classified, and not available for the public.