Implementing an Application to Activate Users during Spare Time in a Military Vehicle

Sofi Westin

December 7, 2009
Master's Thesis in Computing Science, 30 ECTS
Supervisor at CS-UmU: Lena Palmquist
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

The crew in military vehicles occasionally have spare time and they sometimes use external entertainment systems to have something to do. During this time they need to get activated. Research shows that insufficient workload can have a negative effect on performance. The vehicles are equipped with touchscreens that are used to display information and alarms to the crew. If an application can activate the crew, they will have their attention on the screen and easily notice an alarm. This report presents research on workload and attention and also the implementation of the chosen application. The result is a modular application with multiple games that can easily be expanded with other games. The application is incorporated into the new information system that is under development for the military vehicles.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

BAE Systems Hägglunds AB, from now on referred to as Hägglunds, is a company that develops and manufactures military vehicles. One part of the development is the systems that give the crew information about the vehicle and the surroundings. When demonstrating the vehicles for users one question is what there is to do on the screen during spare time. Due to this question and that the answer today is nothing, Hägglunds wanted to implement a prototype that can be used in these situations. The application prototype, later referred to as the application, should activate the user during spare time and it should be possible to motivate, to the buyers, why the application can improve the situation in military vehicles.

1.1 Situation Description

When the crew experiences spare time in the vehicles they can be sitting doing nothing and get bored. To cope with the sometimes long lasting spare time the crew in a vehicle occasionally brings external entertainment systems with them. This can for example be a portable music player or game console. To these they connect their own headphones and doing that they cut themselves off from the vehicle's system. The external entertainment systems are not being controlled by a higher commander. The use of the external entertainment systems also makes it harder to notice an alarm from the systems in the vehicle and also to notice calls on the radio. Another negative aspect of just listening to music or watching a movie is that the crew can become tired and less alert which can affect the work when an alarm or mission occurs.

1.2 Vehicle Equipment

There are many different vehicle types in use today and the equipment in them varies depending on the vehicles' tasks. But some features are the same in all vehicles. All vehicles today have one control system and one information system that give the crew information and feedback on the status of the vehicle. These systems are connected to touch screens located in the vehicle. The touch screens can be found in multiple places in the vehicle. For example, in the SEP vehicles there are five screens in total per vehicle and they are placed accordingly: three in front of the driver, one in front of the commander and one by the role commander, see Figure 1.1. The touch screens can be of different sizes and the control buttons can be placed either on the sides of the screen or below the screen. All touch screens

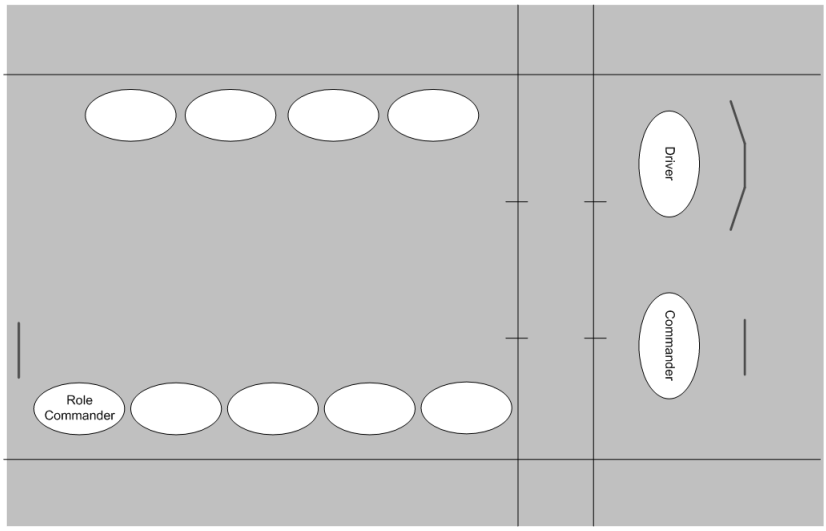
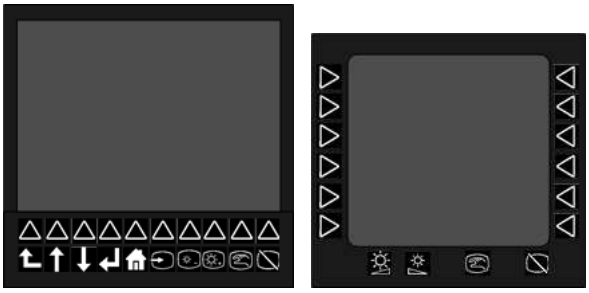


Figure 1.1: An example of the placement of screens in a SEP vehicle



(a) Screen one (b) Screen two

Figure 1.2: Placement of physical buttons on the screens

in one vehicle have the same setting for the control buttons but the size of the screen can differ. The type of screens depends on the kind of vehicle.

The touch screens in the same vehicle are all connected to the same information system but do not depend on each other. The user at each screen can display whatever they like without controlling the other screens. This means for example that the driver can see the driver view while the commander can view the map. Applications in the system can therefore be used differently by each user.

1.2.1 Touch Screen Specification

The touch screens used for displaying the information differ, as mentioned, in size and type. The different sizes can be six, ten and twelve inches. Each screen has physical buttons that can be used to execute commands and are used as backup if the touch function should stop working. These buttons are either placed on both sides of the screen or below the screen, see Figure 1.2. Today the information system is built in a way that all crucial commands can be executed using the physical buttons only.

1.3 Chapter Overview

This report contains the following chapters:

Problem Description This chapter presents the problem, goals and restrictions for the project. It also describes the method used during the work.

Psychological Aspects Here are some arguments described for why this application and this project is important. The chapter describes different aspects on the situation today and why the current situation is not optimal.

Implementation Techniques One part of the specification was to implement the chosen application in the same language and techniques as the information system. This chapter will therefore describe techniques for implementing with Windows Presentation Foundation, eXtensible Application Mark-up Language and Windows Communication Foundation.

Application Description This chapter describes the different parts of the implementation. It contains descriptions of the Graphical User Interface, GUI, development with evaluations and improvements and the other parts of the system. The result is presented here but discussed later in the report.

Conclusions The result is discussed and evaluated against the goals and restrictions set up in the beginning. Some future work and suggested improvements are presented in this chapter.

Appendix The aspects that founded the GUI evaluation is presented here. The chapter Application Description refers to this appendix.

Chapter 2

Problem Description

Hägglunds has become aware of a wish from the users of military vehicles to have an application in the vehicle computers to help them cope with their spare time. The application has to be sellable and therefore it is important to use arguments founded in research for why it is needed.

The information system sends warnings and tasks through the touch screens in the vehicle. When the crew, as today, uses their own headphones and screens this information gets hard to notice. Even if the crew does not use another system they do not give the screen their attention at all times and they can easily miss an alarm. To be able to make the crew to use the built in screens there has to be something interesting to do within the system. This should be some kind of application that is incorporated into the information system so that if an alarm occurs the information system can automatically take over the screen if necessary and show the warning. This leads to increased security when the crew gets the information and with the right kind of application the crew is alert and ready to perform the task at hand.

2.1 Goal Description

The goal of this thesis was to implement an application and incorporate it into the new information system that Hägglunds is developing to replaced the current one. The application should

1. be chosen from the suggestions that the literature study presents
2. have a stand-alone GUI that can be used for different types of applications so that it is easy to incorporate wanted applications
3. have a GUI that has the same look and feel as the entire information system
4. be well fitted with the information system
5. be an application that the crew would want to use
6. keep the crew alert and ready if something would happen
7. activate the crew when there is nothing else to do
8. be implemented as a part of the information system

9. use the same implementation techniques as the information system

The goal with this project is to justify the application to the customer as a selling argument. The research should be able to answer the question: why is this necessary and how does it improve the current situation.

2.2 Restrictions

The application should be incorporated with the new information system and therefore it must be developed using the same programming language as today. The new information system is developed in Windows Presentation Foundation, WPF, and uses Extensible Application Mark-up Language, XAML, for GUI developing. The logic is implemented in c#. Windows Presentation Foundation and Extansible Application Mark-up Language is a part of .NET 3.5 which is a framework developed by Microsoft.

No communication between the screens has been implemented in the new information system yet. There are discussions on what technique and hardware to use for the communication. The application should also follow the same design guidelines and implementation techniques as the information system in order to be perceived as a part of the whole system.

2.3 Method

The approach is to be able to incorporate applications in the information system and use a similar graphical user interface for each application. The information system is built in the .NET environment and the graphical user interface is built with WPF and XAML. The development tool Visual Studio 2008 is used for implementation.

The work began with research on workload and attention to recognize what kind of application to implement. This was done together with a study of the implementation techniques and a learning session of the languages. GUI suggestions were quickly drawn from a picture of a 12 inch screen. The decision on what application to implement was made early in the process and the interaction possibilities in the GUI suggestions was evaluated.

Later after implementing an application, a GUI evaluation was made with the implemented GUI. The logic for the application was developed as a parallel process to the GUI development. The development was made as an iterative process and improved in steps during time. Much information about techniques was found on *msdn/microsoft.com* and in *MSDN Magazine*. With the knowledge that the writers there can be somewhat biased, the information seamed well used by developers. One argument for using the information is that the developers of the language should know how to use it in the best way.

Chapter 3

Psychological Aspects on the Current Situation

During the spare time when the soldiers have nothing to do, they can experience insufficient workload which can have some negative effects on human performance [16]. This is a key argument to motivate the outcoming product to the client. There are two parts that should be satisfied with the product: one is the buyer and the other is the users. The users ask mostly for some kind of game, and the question remains on how to motivate a game to the buyers. Aspects of workload and bordom are well suited for this.

3.1 Workload

When designing a task the workload should be considered. Insufficient workload as well as excessive workload can have a negative effect [14]. Insufficient workload leads to boredom and can result in a deterioration of performance [16]. The workload should not be too high either; that also gives the user problems in performance [9]. In the situation where the crew has spare time it is the insufficient workload that is the problem and when they then need to perform a task they are not alert enough and the performance of the task is not as good as it can be.

Workload can be either mental or physical. Physical workload can for example be how hard it is to physically perform the task. Carrying something really heavy has a higher physical workload than sitting down doing math. On the other hand doing math probably has a higher mental workload than carrying something. In the situation in the military vehicles it is the mental workload that can be affected.

The definition of workload varies, it can for example be defined in terms of effort, demands or as performance [16]. This means that workload can refer to hard work or it can refer to the stress level, or discomfort, that the operator is feeling. It does not have to refer to all aspects at all time. Hard work when it comes to mental workload can be the difficulty of the task. A task can be difficult without causing stress or discomfort and vice versa. For example a time limit can cause stress without any difficulty in the task which then causes a higher mental workload.

It is hard to measure workload due to that it is often subjective. Methods used when measuring workload are often interviews and forms where the user answers questions and grade the workload. When grading the workload it is common to use the Borg scale, see

Table 3.1: BORG-scale for measuring workload

Level	Definition
6	No extertion at all
7	Extremely light
8	
9	Very Light
10	
11	Light
12	
13	Somewhat hard
14	
15	Hard (heavy)
16	
17	Very hard
18	
19	Extremely hard
20	Maximal extertion

Table 3.1. It is a scale from 6 to 20 and when using it for physical workload each number in the scale can represent about one tenth of the users pulse.

Another method that is used is to measure changes in the eyes of the user, for example a size change on the pupil. This method makes it less subjective but different persons react to situations differently, the rating is objective but the reaction is still subjective. It is also a method that few have the possibility to use because of the equipment needed. The most common method is therefore to ask the users and then analyze the answers.

3.1.1 Boredom

All individuals are not necessary likely to experience boredom in the same situation [12]. People react differently to the same situation. This can depend on gender, age, intelligence, and tenure. It shows that younger intelligent men, with relatively high tenure, are more easily bored than others. Research shows that boredom can have a negative effect on task performance. A negative effect can for example mean reduced efficiency performing the task.

3.2 Applications

Today, the crews in the vehicles use first and foremost external entertainment systems during their spare time. Due to this it would be reasonable to guess that the crew wants some kind of entertainment also in the future. The application has to be entertaining enough for the users to choose that one instead of the external systems. This does not mean that the entertainment cannot have a positive effect for the rest of the work. Studies show that computer games can have a positive effect on cognitive performance [7]. One reason for this effect is that the player has to respond and attend to an amount of stimuli in order to

succeed in the game.

Computers and computer playing can have many positive effects on a person if it is used in a good way. It can for example enhance the quality of life and for a child it can improve the social development process [19]. The definition of a good way of using computers and computer playing is: "use it for a suitable amount of time", among other aspects. Social contact with others is the most important feature in an online game [19]. The social contacts comes from helping each other, being part of a group or just play together with others and socialize during the play. Playing together within a group can cause a feeling of belonging and can be seen as a team building exercise.

3.2.1 Content of Games

The content and the type of game have different effects on different people. It has been shown that violent games are positively related to aggressive behavior [10]. This becomes clearer if the player is male or characteristically aggressive. A game with the wrong content can increase aggressive thoughts and behavior. This is an aspect to consider when designing a game for a specific situation. Is increased aggressiveness desireable among soldiers? Women dislike violent games to a greater extent than men but some men also dislike violent games [11].

Games in general need to be challenging and have a clear goal in order to motivate players to continue to play. The instructions should be clear and it should be possible to use a trial and error strategy at the beginning without reading any instruction. This is because many players at first use trial and error and then after a while they may read the instructions [11].

3.3 Attention

Playing games can affect a person's attention and concentration. One study has shown [7] that playing computer games can make a person better at focusing at relevant information. This has been shown when a person plays computer games frequently and it can also increase the attention capacity. These results are supported by more studies according to Barlett et al. [7], where it is shown that playing computer, or video, games can increase the cognitive performance.

Chapter 4

Implementation Techniques

One of the restrictions was to implement the application with the same languages and techniques as the new information system. The information system is implemented in Windows Presentation Foundation, WPF, and eXtensible Application Mark-up Language, XAML. One technique for the communication that is up for discussion is Windows Communication Foundation. These tools were released with .NET 3.5 and are developed by Microsoft Corporation. WPF is an enhancement from a former tool, Windows Forms. WPF makes it easier than it is with Windows Forms to change the look of objects and to make a style of one's own. Microsoft Windows Vista is built with WPF as the foundation. To build a graphical user interface in WPF the declarative mark-up language XAML is used. As with almost all implementation tools guidelines on how to build an application exist. The information system is implemented according to the design pattern Model-View-ViewModel described later. To be able to understand how the implementation has been done for the current information system, knowledge about design guidelines and possibilities the tool provides is necessary. This chapter will therefore present

- Windows Presentation Foundation
- Extensible Application Mark-up Language
- Windows Communication Foudation
- Implementation techniques

4.1 Windows Presentation Foundation

Windows Presentation Foundation supplies different objects to create a graphical user interface, GUI, and can be seen as a library with classes. These objects create a toolbox that can be used for designing an UI. One advantage with WPF, when designing UI:s, is that the WPF application always starts two threads [5]. One thread is available for the developer to control and the other thread WPF is using for rendering and repainting automatically. This means that the developer does not have to care about the rendering and updating the UI when it changes. The rendering uses an instans of the class `DispatcherObject` to queue the operations that should be done [6].

XAML is a declarative mark-up language that WPF uses for creating the graphical user interface. XAML can use the toolbox that WPF provides to design interfaces. In some

```
Button b = new Button();
b.Click += new RoutedEventHandler(button_click);
b.Text = "OK";
```

(a) Imperative programming, example in code behind with c#

```
<Button caption="OK" Click="button_click" />
```

(b) Declarative, example in XAML

Figure 4.1: Example of the difference between imperative and declarative programming

cases though, the designer can use imperative programming in the code behind, written in c# or Visual Basic, VB, to create certain parts of the interface. While using WPF, as much as possible of the interface should be designed in XAML, whereas the logic should be implemented in either c# or VB. Everything implemented in XAML can be implemented in the code behind as well. This means that WPF uses both imperative and declarative programming.

4.2 Declarative Mark-up Languages

Declarative mark-up languages are built in a hierarchical way. There are many different languages in use; the most common are XML and HTML. Declarative programming differs from imperative programming by not requiring algorithms to be run. The code tells only what to accomplish and not how to accomplish it as in imperative programming. This means that the designer does not have to know how to make things happen, he or she just has to know what he or she wants it to become. Declarative programming has a tendency to be more compact and shorter than imperative programming.

Declarative mark-up languages use tags to declare objects. These objects often have properties that can be set within the tag. The content of the object can be declared between the start tag and the end tag. Declarative programming implements objects in a static way and it can be difficult to make changes over time even though in some languages it is possible.

An example of the difference between declarative programming and imperative programming can be seen in Figure 4.1. In the example both codes make the exact same thing, except that in the code behind the parent object has to be specified so that the UI knows where to put the object. In the declarative programming this is defined by the placement of the tag. As shown, imperative programming requires more space and rows than the declarative programming.

4.2.1 Extensible Application Mark-up Language

Extensible application mark-up language, XAML, is a declarative mark-up language that is used with WPF. To be able to use XAML to create a graphical user interface the designer does not have to have any programming skills. The language is built with tags where an object is declared and within the tag properties for the object can be set. Due to the hierarchical nature of the language objects can be declared, and therefore put, inside another object. In this way it is very easy to for example put a picture on a button instead of a just a string with text. XAML is also built in a way that makes it possible and rather easy to change the look of an object. Another advantage about XAML is that it can be split into

```
<Style x:Key="TriggerButton" TargetType="Button">
<Setter Property="Foreground" Value="White">
<Setter Property="Background" Value="Black">
    <Style.Triggers>
    <Trigger Property="IsMouseOver" Value="true">
        <Setter Property="Background" Value="Blue"/>
    </Trigger>
    </Style.Triggers>
</Style>
```

Figure 4.2: Example with trigger in a style

multiple .xaml source files that in the end will be brought together. This makes it possible to use the design in a number of applications without having to rewrite them.

Basic Syntax

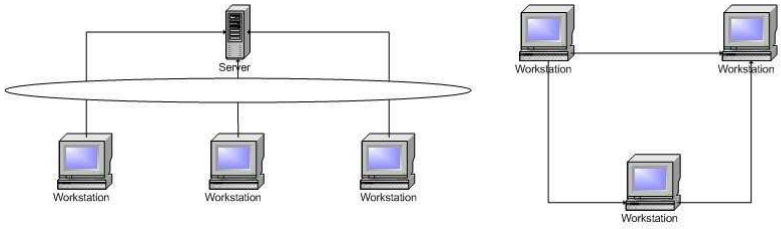
The syntax in XAML is very similar to XML and HTML which makes it easy to learn if knowledge in those languages exists. Every object declaration starts with a '<' and then the type of object that is wanted; for example `<Button>` creates a button. For it to compile the tag has to have an closing bracket '>' and an end tag, the end tag can be written in two ways; either you close the start tag and then use a separate end tag or you write the end tag directly in the start tag. The first way looks like this with the start tag included: `<Button> </Button>`, and the other way: `<Button/>`. The second way takes less space but it takes away the opportunity to put something in between the start tag and the end tag. Every .xaml-file has a code-behind file with extension .xaml.cs. This file is used to handle actions done with the objects created in the .xaml-file. For example, it is possible to specify which method in the code-behind file to evaluate when the user clicks on a button. In the code-behind file it is also possible to create new objects to the interface if wanted, so the designer does not have to use the tag structure for everything.

Advanced Syntax and Resources

As mentioned earlier it is rather easy to customize the look of an object. This is done by creating a resource describing the object. The resource first tells what kind of object it is, it can be compared to inheritance in imperative programming, then it gets a key that is used as a reference name to the resource. Between the resource start and end tag the design part is declared.

To be able to change things on events, XAML have triggers that can be defined and coupled to objects. A trigger can register, for example, when a button is pressed or the mouse is over an object. Triggers define when something should happen and inside the trigger one or more setters are used to define the change or changes that should be performed, see Figure 4.2 for an example. The example in Figure 4.2 shows a simple style that is declared. The declaration takes place in the resource part of the .xaml file. An object can only use one style. It is not possible to combine multiple styles unless a new style is created that combine the other styles.

This style has the key `TriggerButton` and the `TargetType Button`. The `TargetType` says which type of object that can use this style. To use this style on a button the property `Style` has to be set to the key of the style. The example shows a style where a button



(a) Client - server communication (b) Peer to peer communication

Figure 4.3: Illustration of two different communication models

has a black background color and white text and when the mouse is over the button the background color will change to blue.

It is possible to have multiple triggers in the same style and multiple setters in the same trigger. This makes it possible to declare different event happenings in the same style. Styles can be defined within the resources for simple reuse of the style. Resources can also redefine objects as a way to use the object multiple times. A resource can either be dynamic or static. A dynamic resource is evaluated after the loading process when the application is running while a static resource is evaluated during the loading process. Which to use is decided by the implementation. It depends on the type of resource and where it is defined.

4.3 Windows Communication Foundation

Windows Communication Foundation, WCF, is also a part of the .NET 3.5 environment and supplies the designer with components for communication. WCF components support both client-server communication, see Figure 4.3a, and peer-to-peer communication, see Figure 4.3b.

Fundamentally WCF supports message sending between clients and services [3]. Services is an application that waits for messages and responses to them. The clients initiate the communication and sends messages to the services. An application can act as both a service and a client. The important parts to communicate via WCF are an address, a binding, and a contract. An address is needed to create an endpoint to where a message should be sent. A binding is a definition of how the endpoint communicates for example via TCP or HTTP. The binding also tells what security level to use for the transmission of messages. Contracts are used to define how to communicate; operation contracts define parameters and return types for operations, service contracts ties together multiple operation contracts into a single functional unit. It can be compared with an interface with methods where the interface is the service contract and the methods are the operation contracts. In order to have members in the service contract, data contracts are necessary.

4.3.1 Client Side

WCF provides three operation types [15]: first the normal way of calling a method where the client issues a call, blocks while evaluating the method and continues when the method returns. The second way is a one way calling type, where a call is made and then continues without waiting for a response. The third way is duplex callbacks that let the service call back to the client.

Class Name	Transport	Message Encoding	Security Mode
BasicHttpBinding	HTTP	Text	None
WSHttpBinding	HTTP	Text	Message
WSDualHttpBinding	HTTP	Text	Message
WSFederationHttpBinding	HTTP	Text	Message
NetTcpBinding	TCP	Binary	Transport
NetPeerTcpBinding	P2P	Binary	Transport
NetNamedPipeBinding	Named Pipes	Binary	Transport
NetMsmqBinding	MSMQ	Binary	Message
MsmqIntegrationBinding	MSMQ	Not supported	Transport

Table 4.1: Table showing the predefined bindings in WCF

When using one-way operations the client does not care if it was successful or if the call failed. This type is often used in chat applications where a return value is unnecessary and the client can continue writing without any locking when waiting for response. All WCF binding types support a one way communication [15].

WCF provides ten different predefined binding types. These supports different transport protocols and security options. Table 4.1 shows the different types and what each binding supports [20, 21]. In addition to the ones in the table there is one more, namely CustomBinding which has the features the developer decides it should have. It can be customized as a combination of the other bindings to get the wanted features.

To send messages channels are created from the binding. PeerChannel is one type of channel that is created from the NetPeerTcpBinding and it uses multicast when a message is sent, which means that a message from one client is sent to all other clients in the network [17]. With multicast it is best to use a broadcast model to let every other client know the status of the sending client. This can be compared with sending heartbeats to the network. This means also that the communication is a one way communication. To respond to a received message, a new message has to be created and transmitted which leads to a case where all clients receive the response and not only the one it was intended for.

4.4 Implementation Techniques and Design Patterns

Designing and implementing a graphical user interface can be difficult. Common problems can be solved by using design patterns. Design patterns are a description of a problem and the core of the solution.

Microsoft Corporation has developed practices and patterns for designing for WPF [1]. These practices help the developer to create the application in a modular way. The view and the code behind is separated and it is easy to change between views when using these practices. The practices will help a developer to handle a complex application by separating it into smaller parts as modules. Microsoft Corporation recommends developers to use the `Composite Application Library` when designing a client application in WPF. Using the library comes with some benefits [4]; for example simplicity, which means that it reduces the amount of complexity.

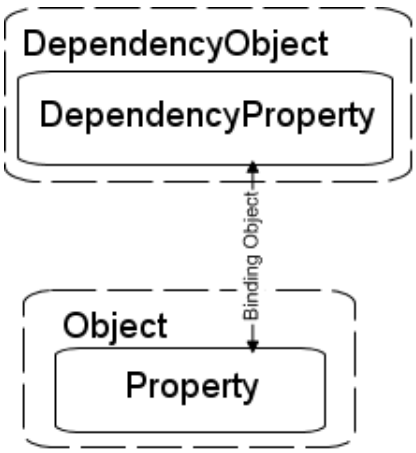


Figure 4.4: A binding between an object property and a DependencyProperty [2]

4.4.1 Data Bindings

WPF has one feature called data binding which, used in the right way, makes the structure less complex. Data bindings make it possible to couple objects, for example a textbox can use a binding for the content so that it is coupled with a variable that changes during the execution of the program. A consequence is that the developer does not have to set the textbox content each time the variable changes. If a variable in the code-behind is coupled to a label presenting data the variable has to be a DependencyProperty. Using DependencyProperties requires the developer to do three things

- Declare and initialize a static readonly DependencyProperty
- Declare a regular property with set and get methods
- Bind the property to the object in the .xaml file

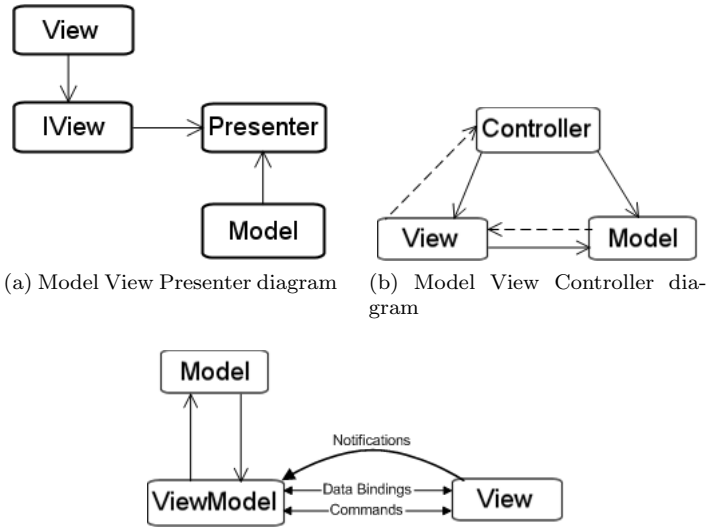
This feature is one of the most useful and advantageous features of WPF. The binding between the property and the label is made using the `Binding` class, see figure 4.4.1 for visualization. The `Binding` works as a connection between the application UI and the business logic [2], which each can be seen as a module in the practices described above.

4.4.2 Design Patterns

Design patterns are architectures that a developer can use as guidelines when implementing an application. The patterns are architectures that define small parts and not complete applications or libraries [13]. There are different categories of design patterns:

- creational
- structural
- behavioral

These categories represent what kind of difficulty the patterns handle. A creational pattern for example defines how objects are instantiated.



(a) Model View Presenter diagram (b) Model View Controller diagram

(c) Model View ViewModel diagram

Figure 4.5: Different design pattern diagrams

There are two ways of implementing a system. One way is to start from a pattern and develop from that. The second way is to start developing the system and let a pattern emerge. If the problem is a common problem the simplest way can be to start from an existing pattern but if the problem is very complex it can be hard to break it down and get it to fit with an existing pattern and therefore the second way can be prefered.

Most patterns are language independent which means that they can be applied in different kinds of languages. Some patterns are specified for object-oriented languages but not a specific one. This is of great advantage when working with many different systems and in multiple languages. There are some exceptions where patterns are made for a specific language, for example a pattern made for WPF is called Model-View-ViewModel and is a popular pattern among developers.

Model-View-ViewModel

Model-Viev-ViewModel, MVVM, is a design pattern tailor-made for WPF [22]. It is very similar to the Model-View-Presenter pattern, MVP, which is a more general pattern. MVP is a variation of the more common Model-View-Controller pattern, MVC.

MVC is a pattern of architectural nature. It separates the presentation from logic which makes it possible to perform tests on each part independently. It also makes maintenance easier for each part. Model-View-Presenter is a variation of this made for .NET-environment. The presenter part does not know about the view at all but uses an interface to control the view [8], see Figure 4.5a. Hence the design can easily be changed without the presenter knowing about it. Also in this pattern the benefit of easy testing follows. In .NET unit testing is a common way of testing and evaluating different parts. Unit testing comes with some rules, one is to isolate the test as much as possible to be able to focus on one specific object [8].

MVVM also separates the view from the logic, see Figure 4.5c. The ViewModel is an abstraction of the view but does not need a reference to the view like MVP does [22]. The

view uses the ViewModel as a DataContext and binds properties to fields in the ViewModel. This makes it a very loosely coupled design which has benefits in maintenance. Another benefit is that a design team can focus on the view and the visual part while a development team can focus on implementing a stable and good ViewModel. It will create a smooth workflow between the teams. Since the design pattern tells how to implement and use the different parts in a specific way, the pattern can be seen as tailor-made for WPF as mentioned before. The pattern depends on the support for commands that WPF provides and would not have been as powerful otherwise.

Due to that the ViewModel does not have to have a reference to the view, the logic can be tested without the view. It is also easy to make different views for GUI evaluations and compare them without changing the ViewModel.

Chapter 5

Application Description

This chapter will present the different parts of the application and describe the final result. First the choice of application is presented and thereafter the system is described.

5.1 Choice of Application

The research done in Chapter 3 concludes that a suitable application can be some kind of game. The game could have some educational value as well but the limitations of the equipment make this a bit harder to solve. Taking in thought that the spare time varies when using the application it should be an application that is easy to interrupt and leave. The ways of interaction are also limited which means that it should not be necessary to have too many different actions and buttons in the GUI. When considering all this the first thing that comes to mind is to make some puzzle games. Puzzle games are many in number and it is easy to interrupt the gaming for another more important task. Popular puzzle games are for example Sudoku, which also demands some logic thinking from the user. Sudoku is a one player game where the user should place numbers in a grid in a certain pattern. It would be beneficial for the crew to have a multi player game to play against or with each other. This can make the crew work better as a team. They are then forced to communicate which can also stimulate them when they are bored. To show how the GUI framework works more than one application should be implemented and to be able to show the communication between different screens at least one multi player application should be implemented. To use the concept of puzzle games Sudoku is chosen as a single player game and for multi player a game called Connect 4 can easily be implemented. Connect 4 is a classic game where the user drops down bricks in a frame to create a row of four bricks in its colour. These games can then be examples of how other applications should be implemented to use the GUI framework.

5.2 System Description

The application is a view of its own in the information system and therefore it should follow the design guidelines used for the rest of the system. This includes both visual and implementation guidelines. The application has to be placed in a specified area on the screen that is decided by the information system, see Figure 5 for illustration. The application contains two parts. One part with the GUI and one part with games using the

Figure 5.1: Shows the available area on a screen for the application

GUI. The UML diagram, see Figure 5.2, shows the building stones of the application. As seen in the Figure 5.2 the games are coupled to the GUI through the interface `IGame`. When a game is chosen an instance of that game is created through the interface. The GUI part is implemented with the MVVM design pattern described in Chapter 4. This makes it easy to change the look of the GUI if wanted. The games and the GUI are somewhat different parts that are implemented separately and then put together in the end.

5.2.1 Graphical User Interface

The development of the graphical user interface, GUI, was made according to an iterative process, see figure 5.3. The iteration started with making two different interfaces, see Figure 5.4a and 5.4b. These were not implemented but only sketched from a picture of the screen. An initial test and discussion about the interaction possibilities gave the GUI with the most buttons an advantage and therefore that GUI was implemented. The most important argument that came up was that it gave a greater freedom from an interaction perspective. All buttons can be altered in terms of content for each game. If a button is not used it is not visible. This creates a possibility to make a GUI almost like the one in Figure 5.4a. The chosen GUI was evaluated in a larger scale to receive more suggestions on improvements.

5.2.2 GUI Evaluation Techniques

Evaluating a GUI is an important part of the iterative design process, see Figure 5.3. It gives the designer feedback and a hint on how to improve the GUI. There are many ways of performing a GUI evaluation. One way is to use a technique called cognitive walkthrough. In a cognitive walkthrough the participants simulate users performing certain tasks [18]. This method is used with experts that have an expertise in the application or in user interface domains. An expert review can be done anywhere in the design process. It is a fast way to get feedback and comments on the system. Often colleagues participate as experts, especially in larger companies. The experts can either give a report after evaluating or they can start a discussion with the designer on the issues they found. Using the discussion technique the designer has a possibility to gather the experts opinions and then explain

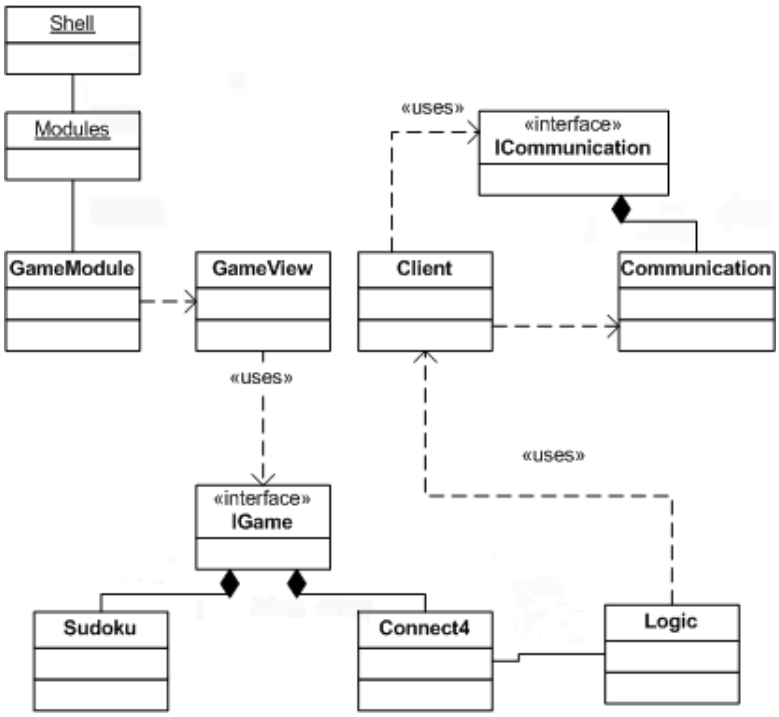


Figure 5.2: UML diagram for the system

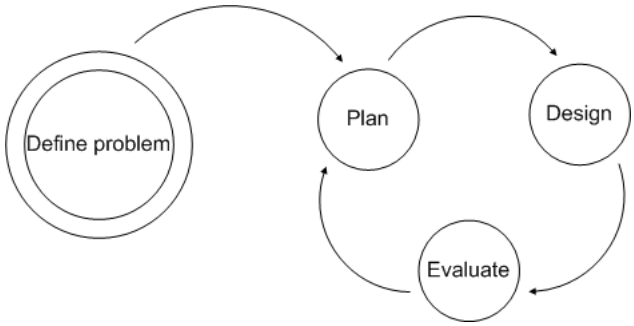
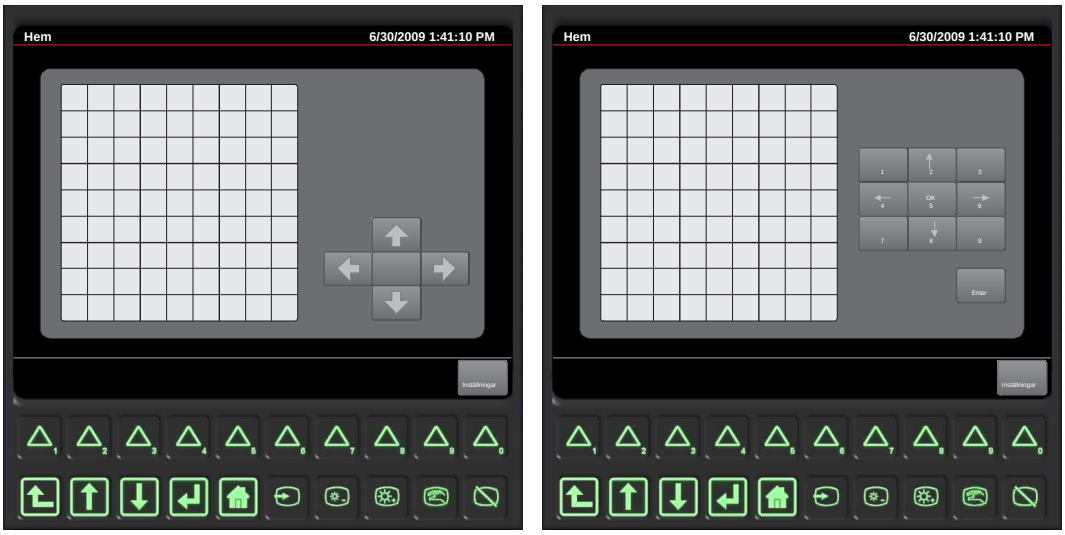


Figure 5.3: An iterative design process



(a) Suggestion one for the GUI (b) Suggestion two for the GUI

Figure 5.4: GUI suggestions with a Sudoku playing field

why it is implemented that way. Some things can be done in a certain way because of the implementation techniques restrict the designer to that way.

5.2.3 Evaluation

The GUI with numbers, see Figure 5.4b, was implemented and after implementing some of the functionality behind it a second evaluation was made. This evaluation started with a pilot study with two persons. The participants made some suggestion for the upcoming evaluation that could give a better result and also save some time for the participants. The pilot study concluded that the right parts were tested and the actual evaluation could start.

The GUI evaluation was performed with 10 participants, chosen from different departments at Hägglunds, and a touch screen was used for a more realistic evaluation. The participants were all coworkers at Hägglunds but have different backgrounds and special fields. Some were programmers, some designers and some were Human-Machine Interaction and Human-Computer Interaction specialists. This gave a wider view of the GUI and the system.

The participants were asked to comment on everything they thought of and use the think-aloud technique. This gives a freedom in what comes up and the participants can say quite different things. The goal was to have as much information as possible and due to the different backgrounds of the participants their comments were expected to vary. Some aspects were prepared, see Appendix A for details, so that all participants would answer them even if they initially did not think of those things. Screen shots from the evaluation can be seen in Figure 5.5.

5.2.4 Evaluation Results

The comments on the different parts were put together and resulted as follows:

Buttons

The emphasized parts during the evaluation regarding the buttons were their usage and visual aspects. The participants all agreed that the position and appearance was good. However a button should be inactivated in situations where no function is available. For example should the `correct` button in the Sudoku game be disabled until the grid is completely filled. It is unnecessary to correct the grid before that since it is obviously incorrect. No participant complained about the size, some expressed that the size was good.

Menu

Regarding the menu, all test participants agreed on that `settings` was not a suitable name on the menu button. A suggestion was to change it to `menu`. There were discussions whether `new game` should be in the menu or be a stand alone button on the screen. A stand alone button would have to be combined with a popup question asking if the user is sure. Another suggestion was to be able to choose different games directly with buttons without traversing a menu.

Sudoku

The input in Sudoku worked really well during the tests. All participants figured out for themselves how to do it. Although one participant wanted to choose a number first and then the square to put it in, the participant also commented on that it was a habit from somewhere else and that it worked fine to choose the square first, when the participant became used to it. The main issue was the fact that it was not really clear which numbers were predefined and not. There should be a greater difference, all participants said. The predefined numbers should be disabled too so that it is really clear that the user cannot change them.

Most test participants asked for difficulty levels to choose from. Some participants that were not really used to playing Sudoku wished for hints for a next move. This could be to point out one square that should be easy to solve or check when placing a number if it is obviously incorrect. The hints should then be able to turn on and off. Another thing that was noticed was that the participants did not realize the meaning of the button Done. This button was used to correct the Sudoku but this was not clear. It should therefore change name to correct or accordingly. It should be possible to start over with the current game if it shows to be incorrect what is written and also when going to another view the current game should be saved so that when the next break comes the user can continue where he left off.

Some additional things to make it more fun is to have a highscore within the vehicle so that players can compete on who solved it the fastest. Some more colours could also be fun to lighten up the game a little.

Connect 4

All logic behind the game was not working and therefore the participants could only try making one move and then pretending to wait for an opponent to make their move. The row above the grid should be more clear so that the user understands what to do. There were several comments on how to input. For example a wrap-around function was wished for so that when the user presses left when standing as far as possible to the left the ball moves to the place as far to the right as possible. Some participants wanted to have a quicker way to

choose a column. One suggestion was to click on the column and then arrow down, another was to double-click on the column. One suggestion was to have buttons below the playing area where the user chooses the column. One aspect that opposes these suggestions is the wish to not have to place the hands in the playing area.

Once someone has won the winning row should be marked for the players to see. Also the colour of the players should not be red and green because of colour blindness. Suggestions said yellow and red or blue and red. Some participants wished for a more graphical look on the game, perhaps some 3D graphics and a more fun background. A common suggestion was an animation when a ball drops down in a column.

Other

Comments were made on the graphics, both that it sufficed with simple graphics but also opinions about having more fun graphics and a more colourful playing area. An argument for it was that it would perhaps help the users to relax and forget the surroundings a bit.

5.2.5 Conclusion of Evaluation

On some aspect the participants agreed but on some points the opinions differed. Most of the GUI worked well. However some points of improvement and how to clarify the interaction was made. The things that should be changed were

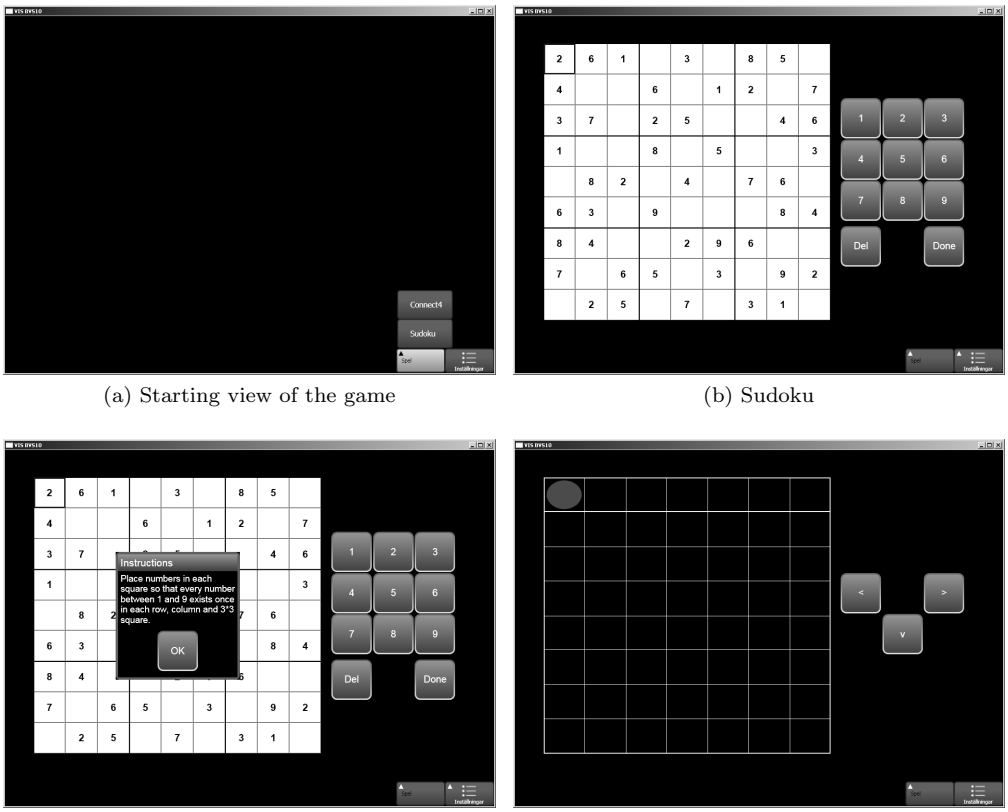
1. Change name on the button `Done` in Sudoku to `Correct`
2. Change colours in Connect 4 due to colour blindness
3. Use more graphics in the games
4. Disable the button `Done` in Sudoku until the whole grid is filled with numbers
5. Implement a restart function in Sudoku
6. Implement a help popup in Connect 4
7. Implement a faster interaction possibility in Connect 4

The things was prioritised after how the participant agreed on it and by the complexity in implementation. If the participants disagreed the aspect got a lower priority.

5.2.6 Game Implementation Description

All games need to implement the interface `IGame`. This provides the game with control over the buttons' contents and the menu in the lower right corner. Every field in the interface is read directly after creating the instance of the game and if something changes in the GUI during the game an update event has to be created. The button's content uses the technique of dependencyProperties that notifies the DispatcherObject if something changes. If an update event is made the dependencyProperties will be updated. The .xaml class has to be declared as a UserControl object so that it can be used in the playing area.

The Connect 4 game is a two player game which means that it has to communicate with another screen in the vehicle. This is made by using WCF. The communication is implemented as a separate part of the information system. The client is implemented by the singleton design pattern which supports other parts using it as well. The communication is a peer-to-peer solution and all messages are sent with an identifier of the sender.



(a) Starting view of the game

(b) Sudoku

(c) Sudoku with help popup

(d) Connect4

Figure 5.5: Screenshots from GUI evaluation

When a message arrives to the system an event is created and the application using the communication channel has to listen to these events. The message is sent to all systems that are connected within the network, hence a game with more than two players can easily be implemented.

Chapter 6

Conclusions

The aim of this thesis was to develop a prototype that worked on at least one of the touch screens. The prototype would have multiple games in it to show how the GUI framework can be used. The aim was to fulfill the goals and develop an application that felt usable.

During the work there has been some problems occuring and the time plan has been edited along the way. The result is almost what was expected before starting this work but with the exception that some goals is not really reached. This is due to that the tools took longer than expected to learn and also the code for the current information system was complex and hard to understand at the beginning. The goal regarding the justification can be seen as fulfilled founded in research and scientific reports. This was an important part for BAE Systems Hägglunds to have and it was an interesting part to do.

The goal of having a stand-alone GUI is fulfilled in some sense. It can be implemented more generically but the purpose is fulfilled. The purpose was to make it easy to create new applications and use them in the system. Making the GUI stand-alone took longer than expected and the GUI evaluations were delayed. The consequence was that there was less time to implement the suggested changes from the evaluations.

To be able to show the GUI it was necessary to have more than one application implemented. The Sudoku was the first to be implemented. That application went quite nicely and the second application, Connect 4, was implemented next. Due to that Connect 4 is a two player game a communication had to be set up. This caused many problems. Some vehicles have a server-client connection and some have a peer-to-peer connection. The goal when discussing the communication part was to be able to easily control which connection to use. I decided to start with the peer-to-peer solution which came with a lot of problems. Most problems were about firewalls and Windows components. These things made the application to throw exceptions on a lower level which made the problem hard to solve.

6.1 Resulting Application

The result is an application in three layers. One communication layer that can be used by every module in the information system. One layer is the game module with the GUI framework. The third layer is the implemented games and connected to the game module through the interface `IGame`. The application works with the information system and can be motivated by the research described in Chapter 3.

The project ended with a product that is almost ready to use. There are some limitations and restrictions to the product. These should be erased before the product can be used.

6.2 Restrictions

The application is implemented with the Model-View-ViewModel pattern as the information system is today. The GUI follows this but looking at the next level with the games these are connected to the GUI by using the Model-View-Presenter pattern. In this way it was seperated from the entire system and it can be seen as an individual system inside the information system. The change between games could be made easier if the application only uses Model-View-ViewModel due to that the pattern makes it easy to change views.

Implementing the communication was one of the last things conducted in the study and the peer-to-peer solution took longer than expected. Due to the troubles with firewalls and Windows components there was not enough time to implement a working the client-server solution. The important thing was to get one connection to work as it should.

6.3 Limitations

There are some limitation in the GUI. For example it is only possible to place strings in the button contents though it should be able to handle all kinds of UI elements. The control over the buttons is also hard to solve. There is no chance for the different games to control if the button for example should be disabled. Either the buttons are invisible, which happens when no content is set, or the buttons are visible and enable. The complete application also needs some testing for bugs and other security aspects. I have not yet performed a real stability test on the application to see what happens in different situations when the user makes something unexpected.

6.4 Future Work

The application is built in a way that it always can be expanded and never be found completely done. For example more games and other applications can be implemented so that the crew have more choices. Only seeing to the implemented parts a lot can be done in the future. The Sudoku game could be expanded with levels and hints. The graphic part can be rebuilt to make it more fun if that is what the users want. There are many functions that can be added to that kind of game.

Connect 4 can be expanded to handle more than two players and the graphical part can be changed to a more colourful and playful game. Otherwise this game is pretty completed and cannot be developed further so much more due to the simplicity of the game concept.

For the future the communication between the systems should be expanded to handle both client-server and peer-to-peer solutions. One parameter should control which one to use. This is one important feature if the system should be used in different types of vehicles.

Chapter 7

Acknowledgements

I want to thank my external supervisor Karin Fossum and all other people who has helped me in my work at BAE Systems Hägglunds. A special thanks to the participants in the GUI evaluation and the development team for the information system.

I also want to thank my internal supervisor Lena Palmquist for the help with the report and Per Lindström for the guidance in the beginning of the thesis work.

References

[1] Composite Application Guidance for WPF and Silverlight. http://msdn/microsoft.com/en-us/library(d=default)/dd458809(l=en-us,v=MSDN.10).aspx, February 2009.

[2] Data Binding Overview. http://msdn/microsoft.com/en-us/library(d=default)/ms752347(l=en-us,v=VS.90).aspx, October 15 2009.

[3] Fundamental Windows Communication Foundation Concepts. http://msdn/microsoft.com/en-us/library(d=default)/ms731079(l=en-us,v=VS.90).aspx, September 30 2009.

[4] Goals and Benefits. http://msdn.microsoft.com/en-us/library(d=default)/dd458906(l=en-us,v=MSDN.10).aspx, February 2009.

[5] Threading Model. http://msdn/microsoft.com/en-us/library(d=default)/ms741870(l=en-us,v=VS.90).aspx, October 20 2009.

[6] Windows Presentation Foundation. http://en.wikipedia.org/wiki/Windows_Presentation_Foundation, August 31 2009.

[7] Christopher P. Barlett, Christopher L. Vowels, James Shanteau, Janis Crow, and Tiffany Miller. The effect of violent and non-violent computer games on cognitive performance. *Computers in Human Behavior*, 25:96–102, 2009.

[8] Jean-Paul Boodhoo. Design Patterns: Model View Presenter. *MSDN Magazine*, August 2006.

[9] Committee on the Effect of Smaller Crews on Maritime Safety, National Research Council. *Crew size and maritime safety.* The National Academies Press, 1990.

[10] Karen E. Dill Craig A. Anderson. Video games and aggressive thoughts, feelings, and behavior in the laboratory and in life. *Journal of personality and social psychology*, 78(4):772–790, 2000.

[11] John V. Dempsey, Linda L. Haynes, Barbara A. Lucassen, and Maryann S. Casey. Forty simple computer games and what they could mean to educators. *Simulation and gaming*, 33(2):157–168, 2009.

[12] Annilee M. Game. Workplace boredom coping: health, safety, and HR implications. *Personnel Review*, 36(5):701–721, 2007.

[13] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Abstraction and Reuse of Object-Oriented Design. In *ECOOP '93 - Object-Oriented Programming*, volume 707 of *Lecture Notes in Computer Science*, 1993.

[14] Gavriel Salvendy John R. Cook. Job enrichment and mental workload in computer-based work: Implications for adaptive job design. *International Journal of Industrial Ergonomics*, 24:13–23, 1999.

[15] Juval Lowy. What You Need To Know About One-Way Calls, Callbacks, And Events. *MSDN Magazine*, October 2006.

[16] Ministry of Defence, Defence Standard 00-25. *Human Factors for Designers of Systems, Part 16: Introduction and Manpower Domain*, 1 edition, July 2004.

[17] Matt Neely. A Peer-To-Peer Work Processing App With WCF. *MSDN Magazine*, June 2009.

[18] Ben Schneiderman and Catherine Plaisant. *Designing the user interface.* Pearson Education, Inc., 4 edition, 2005.

[19] Qingxin Shi. Why some people are addicted to computer games - an analysis of psychological aspects of game players and games. Master's thesis, Department of informatics, Copenhagen Business School.

[20] Aaron Skonnard. Learn The ABCs Of Programming Windows Communication Foundation. *MSDN Magazine*, February 2006.

[21] Aaron Skonnard. WCF Bindings In Depth. *MSDN Magazine*, July 2007.

[22] Josh Smith. WPF Apps With The Model-View-ViewModel Design Pattern. *MSDN Magazine*, February 2009.

Appendix A

GUI Evaluation Aspects

The prepared aspect for the GUI evaluation were

- Buttons:
    1. Position
    2. Size
    3. Visual
    4. Visibility
- Menu:
    1. Structure
    2. Belonging
- Sudoku game:
    1. User friendliness
    2. Instructions
    3. Way of correcting
    4. Restart the game
- Connect 4 game:
    1. Interaction way
    2. Is the top row clear enough?
- Other aspects
    1. Question when changing game
    2. Language on instructions and menu